

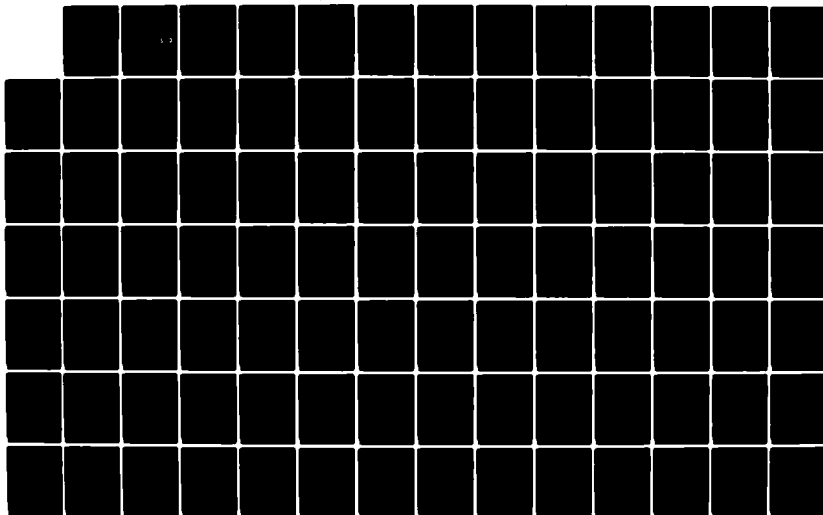
AD-A145 757

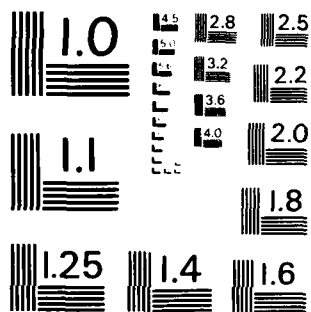
A METHODOLOGY FOR THE ANALYSIS OF PROGRAMMER
PRODUCTIVITY AND EFFORT ESTI..(U) AIR FORCE INST OF
TECH WRIGHT-PATTERSON AFB OH J D FERNANDEZ MAY 84
AFIT/CI/NR-84-44D F/G 9/2

1/3

UNCLASSIFIED

NL





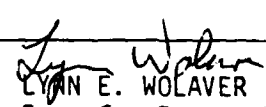
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A145 757

DTIC FILE COPY

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|--|---|--|
| 1. REPORT NUMBER AFIT/CI/NR 84-44D | 2. GOVT ACCESSION NO. AD-A145 757 <i>PRODUCTIVITY</i> | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) A Methodology For The Analysis Of Programmer And Effort Estimation Within The Framework Of Software Conversion | | 5. TYPE OF REPORT & PERIOD COVERED THESIS/DISSERTATION |
| 7. AUTHOR(s) John D. Fernandez | | 6. PERFORMING ORG. REPORT NUMBER |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Texas A&M University | | 8. CONTRACT OR GRANT NUMBER(s) |
| 11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | | 12. REPORT DATE 220 |
| | | 13. NUMBER OF PAGES 1984 |
| | | 15. SECURITY CLASS. (of this report) UNCLASS |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |
| 16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED | | |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) B | | |
| 18. SUPPLEMENTARY NOTES APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1/ | | |
| <div style="text-align: right;">  LYNN E. WOLAVER Dean for Research and Professional Development AFIT, Wright-Patterson AFB OH </div> | | |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number) | | |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number) ATTACHED | | |

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

84 09 13 007

ABSTRACT

A Methodology For The Analysis Of Programmer Productivity And Effort
Estimation Within The Framework of Software Conversion (May 1984)

John Diego Fernandez, B.A., Texas A&I University;
M.S.E., West Virginia University

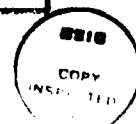
Software conversion is becoming increasingly significant as the inventory of programs increases and as the life cycle of many applications becomes longer. Although some work has been done in the area of software conversion, it has received little research attention since it has only recently become a more frequent occurrence. This research considered two aspects of software conversion and developed a methodology for the statistical analysis of conversion sample data from the ongoing U.S. Air Force Base Level Data Automation Program (officially given the short title Phase IV).

The two areas specifically addressed by this research were programmer productivity and effort estimation. Programmer attributes and program characteristics were studied in relation to programmer productivity in software conversion. Models for explaining productivity were constructed and the impact of organization was also considered. Existing applicable models for software conversion effort estimation were examined and their accuracy was evaluated.

Environment specific regression models for effort estimation were also constructed.

Several statistical and summarizing techniques were considered for the analysis of the conversion sample data. As various aspects of the data were studied, selected statistical techniques emerged as more appropriate. These provided the basis for the methodology formulated and used throughout the data analysis. The Air Force data was utilized in a case study of the application of the methodology. The analysis of conversion programmer productivity revealed that experience, lines of code, a programmer's knowledge of the program, organization, and other factors and attributes impacted productivity. Of the effort estimation models studied, the Hahn and Stone model exhibited the best performance while the Federal Conversion Support Center model had the lowest accuracy.

| | |
|--------------------|--|
| Accession For | |
| NTIS GRA&I | <input checked="checked" type="checkbox"/> |
| DTIC TAB | <input type="checkbox"/> |
| Unannounced | <input type="checkbox"/> |
| Justification | |
| PER CALL JC | |
| By | |
| Distribution/ | |
| Availability Codes | |
| Dist | Avail and/or Special |
| A-1 | |



AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (AU). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

PRODUCTIVITY

RESEARCH TITLE: A Methodology For The Analysis Of Programmer And Effort Estimation Within The Framework Of Software Conversion

AUTHOR: John D. Fernandez

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?
☐ a. YES ☐ b. NO
2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?
☐ a. YES ☐ b. NO
3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?
☐ a. MAN-YEARS _____ ☐ b. \$ _____
4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?
☐ a. HIGHLY SIGNIFICANT ☐ b. SIGNIFICANT ☐ c. SLIGHTLY SIGNIFICANT ☐ d. OF NO SIGNIFICANCE
5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME _____

GRADE _____

POSITION _____

ORGANIZATION _____

LOCATION _____

STATEMENT(s): _____

A METHODOLOGY FOR THE ANALYSIS OF PROGRAMMER PRODUCTIVITY AND EFFORT
ESTIMATION WITHIN THE FRAMEWORK OF SOFTWARE CONVERSION

A Dissertation

by

JOHN DIEGO FERNANDEZ

Submitted to the Graduate College of
Texas A&M University
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

May 1984

Major Subject: Computer Science

84 09 13 007

A METHODOLOGY FOR THE ANALYSIS OF PROGRAMMER PRODUCTIVITY AND EFFORT
ESTIMATION WITHIN THE FRAMEWORK OF SOFTWARE CONVERSION

A Dissertation

by

JOHN DIEGO FERNANDEZ

Approved as to style and content by:

Sallie Sheppard
S.V. Sheppard
(Chairman)

W.M. Lively
W.M. Lively
(Member)

William Luent
W.L. Fuerst
(Member)

D.B. Simmons
D.B. Simmons
(Member)

Bruce H. McCormick
B.H. McCormick
(Head of Department)

May 1984

ABSTRACT

A Methodology For The Analysis Of Programmer Productivity And Effort
Estimation Within The Framework of Software Conversion (May 1984)

John Diego Fernandez, B.A., Texas A&I University;

M.S.E., West Virginia University

Chairman of Advisory Committee: Dr. S.V. Sheppard

Software conversion is becoming increasingly significant as the inventory of programs increases and as the life cycle of many applications becomes longer. Although some work has been done in the area of software conversion, it has received little research attention since it has only recently become a more frequent occurrence. This research considered two aspects of software conversion and developed a methodology for the statistical analysis of conversion sample data from the ongoing U.S. Air Force Base Level Data Automation Program (officially given the short title Phase IV).

The two areas specifically addressed by this research were programmer productivity and effort estimation. Programmer attributes and program characteristics were studied in relation to programmer productivity in software conversion. Models for explaining productivity were constructed and the impact of organization was also considered. Existing applicable models for software conversion effort estimation were examined and their accuracy was evaluated.

Environment specific regression models for effort estimation were also constructed.

Several statistical and summarizing techniques were considered for the analysis of the conversion sample data. As various aspects of the data were studied, selected statistical techniques emerged as more appropriate. These provided the basis for the methodology formulated and used throughout the data analysis. The Air Force data was utilized in a case study of the application of the methodology. Unique features of the data which entered the analysis included government programmers, less than half being college graduates, working in different organizations and converting COBOL-68 programs to COBOL-74 with the maximum program length being 5000 lines. The analysis of conversion programmer productivity revealed that experience, lines of code, a programmer's knowledge of the program, organization, and other factors and attributes impacted productivity. In addition, the Hahn and Stone effort estimation model exhibited the best performance in estimating system level effort while the Federal Conversion Support Center model had the lowest accuracy. Other effort models were also studied.

ACKNOWLEDGEMENTS

First, I thank my wife Mary and our children, Daniel and Monica, for their love and understanding and for getting along without me for much of the time that I worked on this project. I then acknowledge my chairperson Dr. Sallie V. Sheppard who became a trusted friend and whose encouragement and confidence helped to sustain me through the rough spots of this endeavor. I thank Dr. W.M. Lively and Dr. D.B. Simmons for their support and for helping to adjust the focus of my research objectives. Dr. W.L. Fuerst provided a welcomed broader perspective. Dr. R.P. Schmitt, my Graduate College Representative, was always very supportive and readily available. Mr. Eddy Fernandez, C.D.P., suggested some ideas to pursue for more efficient data entry. 1Lt J. Cogburn and 2Lt R. Naylor, of the Air Force Automated Systems Project Office(AFASPO), provided the data and invaluable support for this research. I especially thank Col. R. Hedges who approved the release of the data. Four other Air Force organizations, AFSDC, TAC, SAC and ATC, provided additional information. Capt Ken Hebert's experience with Phase IV prompted my investigation of the conversion area. I thank Anne Coleman from the Statistics Department for her guidance during the research and for reviewing the manuscript.

There are many individuals at Texas A&M University, with whom I came in contact, that made my time here a very worthwhile and rewarding experience.

Last but not at all least, I thank the Lord for being there!

TABLE OF CONTENTS

| CHAPTER | Page |
|---|------|
| I INTRODUCTION | 1 |
| Description of the Conversion Problem | 2 |
| Software Conversion Fundamentals | 4 |
| Current Research Interest in Conversion | 6 |
| Air Force Base Level Data Automation Program | 8 |
| Overview of Research and Chapter Summary | 9 |
| II PANORAMIC VIEW OF SOFTWARE CONVERSION | 11 |
| Conversion Types and Techniques | 11 |
| Conversion Process | 17 |
| Conversion Planning | 19 |
| Management Concerns | 22 |
| The Contracting Option | 25 |
| Conversion Tools | 25 |
| III CONVERSION EFFORT ESTIMATION AND PRODUCTIVITY | 28 |
| Cost/Effort Estimation | 29 |
| Basic Estimating Concepts | 30 |
| Conversion Effort/Cost Estimation Models | 32 |
| Development Models of Interest | 35 |
| Software Conversion Productivity | 37 |
| Productivity Measurements | 37 |
| Productivity Studies | 40 |
| IV AIR FORCE PHASE IV PROGRAM | 51 |
| Program Prescription | 51 |
| Phase IV Materialization | 53 |
| Conversion Assistance, Tools and Procedures | 54 |
| Initial Conversion Experiences | 57 |
| Conversion Effort Data Collection | 59 |
| V CONVERSION PROGRAMMER PRODUCTIVITY ANALYSIS | 66 |
| Introduction | 66 |
| Preliminary Analysis | 66 |
| Definitions and Assumptions | 67 |
| General Overview of Analysis Methodology | 68 |
| Selection of Relevant Variables | 69 |

Table of Contents (Continued)

| CHAPTER | | Page |
|---------|--|------|
| | Categorical Variables Subjected to Analysis of | |
| | Variance | 69 |
| | Continuous Variables Scrutinized | 74 |
| | Model Variables Selected | 76 |
| | Model Specification and Analysis | 76 |
| | Initial Model Analysis | 77 |
| | Reduction of Initial Model | 79 |
| | Alternate Dependent Variable Models | 84 |
| | Consideration of Organizational Impact | 88 |
| | Model Validation | 90 |
| VI | SOFTWARE CONVERSION EFFORT ESTIMATION ANALYSIS | 94 |
| | Introduction | 94 |
| | Respecification of Effort Estimation Models | 96 |
| | FCSC Cost Model | 97 |
| | Hahn and Stone or MITRE Model | 101 |
| | Grim, Epler and Andrus Model | 103 |
| | Wolberg Model | 104 |
| | Basili and Freburger Model | 105 |
| | Validation of Existing Models | 106 |
| | Measurement of Accuracy of Basic Models | 106 |
| | Analysis of Refined Models | 108 |
| | Development of Models With Regression Analysis | 112 |
| | Exponential Form Effort Model | 113 |
| | Additive Form Effort Model | 114 |
| | Final Comparison of Models | 115 |
| | Organizational Impact Model | 117 |
| VII | SUMMARY AND RECOMMENDATIONS | 120 |
| | Introduction | 120 |
| | Overview of Work Accomplished | 120 |
| | Significance of Research Outcomes | 121 |
| | Summary of Methodology Formulated | 121 |
| | Productivity Methodology | 123 |
| | Effort Estimation Methodology | 124 |
| | Summary of Productivity Analysis | 126 |
| | Summary of Effort Estimation Analysis | 129 |
| | Management Considerations | 131 |
| | Data Collection Forms | 131 |
| | Data Submission Procedures | 132 |
| | Controlling the Process | 133 |
| | Personnel Selection Considerations | 134 |
| | Future Research Possibilities | 134 |

Table of Contents (Continued)

| CHAPTER | Page |
|---|------|
| REFERENCES | 137 |
| GLOSSARY | 143 |
| APPENDIX | |
| A DETAILS OF CONVERSION EFFORT/COST ESTIMATION MODELS | 149 |
| B DATA ENCODING AND PRELIMINARY ANALYSIS | 176 |
| C PROGRAMMER RESUME FILE | 199 |
| D PROGRAM INFORMATION FILE | 209 |
| VITA | 220 |

LIST OF TABLES

| TABLE | Page |
|---|------|
| 1 Classification of Conversion Efforts | 12 |
| 2 Comparison of Phases of Software Conversion | 18 |
| 3 Lines of Code Per Hour(LOC PER HR) Averages for Categorical Variables. | 71 |
| 4 Initial Version of Productivity(LOC PER HR) Model. | 78 |
| 5 Final Version of LOC PER HR Model. | 81 |
| 6 Final HRP ERH LO Alternate Productivity Model. | 85 |
| 7 Final LOG LOCPH Alternate Productivity Model. | 87 |
| 8 Final LOC PER HR Model With Organization. | 89 |
| 9 Phase IV System Level Effort Data. | 95 |
| 10 Summary of Conversion Effort Estimation Models. | 107 |
| 11 Validation/Accuracy Measures of Basic Models. | 107 |
| 12 Organizational Impact Effort Model | 118 |
| 13 Task Percentages for FCSC Complexity Classes | 152 |
| 14 Hahn and Stone Conversion Production Mean Rates | 165 |
| 15 Hahn and Stone Documentation Status Categories | 165 |
| 16 Hahn and Stone Modification Level Ratings | 166 |
| 17 College Education Categories. | 177 |
| 18 Academic Majors and Minors. | 178 |
| 19 Formal Training Categories. | 179 |
| 20 Programmer Experience Categories. | 179 |
| 21 Programming Language of Programs to Convert. | 180 |

List of Tables (Continued)

| TABLE | Page |
|--|------|
| 22 Conversion Experience Categories. | 181 |
| 23 Programmer Resume Data Record. | 182 |
| 24 Program Information Data Record. | 183 |
| 25 Types of Phase IV Programmers. | 185 |
| 26 College Education of Phase IV Programmers. | 185 |
| 27 Summary of Majors of Phase IV Programmers. | 186 |
| 28 Formal Training Profile of Phase IV Programmers. | 186 |
| 29 Conversion Experience of Phase IV Programmers. | 187 |
| 30 Regrouping College Education, Major and Conversion Categories. | 189 |
| 31 Summary of Chi-Square Tests. | 190 |
| 32 Partial Correlation Matrix and Factor Analysis. | 191 |
| 33 Phase IV Programmers Experience Summary. | 192 |
| 34 Phase IV Programs By Type and Number of Programmers. | 193 |
| 35 Program Difficulty Counts & Totals. | 194 |
| 36 Conversion Activities: Times & Percentages. | 194 |
| 37 Productivity and Other Summary Measures. | 195 |

LIST OF FIGURES

| FIGURE | Page |
|--|------|
| 1 AFASPO Phase IV Programmer Resume Form | 61 |
| 2 AFASPO Phase IV Program Information Form | 62 |
| 3 Plots of Estimates of Basic Models for a Small System . . . | 109 |
| 4 Plot of Refined Models and Regression Developed Models . . . | 116 |
| 5 LOCPERHR versus Knowledge(KCA) of programmer for single programmer type programs. | 197 |
| 6 LOCPERHR versus Program Difficulty(SUMDIF) for single programmer type programs. | 198 |

CHAPTER I

INTRODUCTION

Software conversion is becoming increasingly significant as the inventory of programs increases and as the life cycle of many of these applications becomes longer. Although some work has been done in the area of software conversion, it has received little research attention since it has only recently become a more frequent and costly occurrence. Additionally, little software conversion effort data has been available outside the few vendor firms performing conversions. In the field, conversion has not received the attention it merits since it has traditionally been considered to be of "low status".

This research developed a methodology for the analysis of software conversion which investigates programmer productivity and effort estimation. Two items specifically studied are the accuracy of software conversion effort/cost estimation models and the correlation between conversion programmer productivity and programmer attributes. The procedures developed for the analysis were used in a case study of conversion sample data available from the ongoing Air Force Base Level Data Automation Program (officially short-titled Phase IV) which involves the replacement of over 200 computer hardware systems and the associated conversion of about 300 software

The journal used as a pattern for format and style was Computing Surveys.

systems by over 20 different Air Force organizations. Environment-specific effort estimation and productivity models resulted from the analysis. The overall results are primarily applicable to one type of conversion but provide guidance and a foundation for work with all types.

Description of the Conversion Problem

The U.S. General Accounting Office(GAO) has found that the length of the life cycle of computer hardware systems within the federal government is about seven years[General Accounting Office, 1977]. This infers that every federal organization using a computer may sooner or later have to convert its application software. By increasing the capacity of the existing hardware configuration, an organization may find it possible to postpone a conversion. However, sooner or later, every organization will have to replace its computer system with a new machine for technical and/or political reasons.

Wolberg sees the need for conversion as being dependent on the failure of a particular computer environment to function as required or because a vendor has discontinued support for a specific piece of software or hardware[Wolberg, 1983]. Changes become necessary for one or more of the following incentives: increased capacity, increased reliability, improved performance or reduced cost. Whatever the reason for the change, an analysis of the costs and available options and impacts must always take place. Some changes, such as the acquisition of additional disk drives, rarely affect

software. However, the replacement of one computer with another will typically require some degree of changes to the software especially if the architecture changes.

The GAO reported in 1977 that the federal government was spending more than \$450 million per year to convert programs[General Accounting Office, 1977]. Though conversions may be more frequent within the government, Wolberg stated in 1983, that the worldwide annual cost of conversion was estimated at several billion dollars and that surveys estimated the cost of conversion to be as high as 10% of the computing budget[Wolberg, 1983]. Boehm stated that the annual cost of software in the U.S. in 1980 was approximately 40 billion dollars or about 2% of the Gross National Product(GNP)[Boehm, 1981]. He further added that these costs are expected to grow to 8.5% of the GNP by 1985 and 13.5% by 1990. By analyzing these cost statements, one can conclude that a current estimate of annual software conversion costs in the U.S. may be about \$4 billion.

The GAO report includes the results and analysis of a GAO survey which concluded that about 24% of software conversion costs could be eliminated by improving the conversion process as well as the quality of new software development which will make future conversions easier. As the cost of software increases so does the importance of producing cost effective software through development and conversion. The challenge to software professionals is quite clear[Fernandez, 1982]. Software conversion research efforts are essential to the industry.

Boehm emphasized that poor management can increase software costs more rapidly than any other factor[Boehm, 1981]. This statement was made concerning new software development, but it is also applicable to conversions. Oliver pointed out that most of the ignorance regarding conversion has to do with the process itself[Oliver, 1978]. All too often, organizations mistakenly liken conversion to development, fail to plan and prepare properly, and invariably allocate resources parsimoniously to the conversion effort. Planning for the software conversion required in a hardware replacement is usually done too late to avoid multiplying the problems of a conversion.

Software Conversion Fundamentals

The need or desire to move software from one environment to another is fundamental in the usage of computers. This move typically occurs during the maintenance phase of an application system's life cycle and it could be triggered by a new operating system, a new hardware configuration, or language and compiler changes. Wolberg suggested three basic alternatives that may be considered when such a move is contemplated[Wolberg, 1983]:

1. Emulation -- A process by which the new environment is made to directly execute software written for the original environment.
2. Conversion -- A process in which changes are made to the software so that the original system will execute properly in the new environment.

3. Replacement -- Alternative software is either developed or acquired for the new environment. This is the most radical choice.

D. Schneider suggested a fourth alternative that could be considered[Schneider, 1978]:

4. Termination -- Considering termination forces the examination of the essentiality of the system.

If emulation is possible, no software changes are necessary. However, emulation is not a feasible alternative because it is often costlier than imagined and is least effective in terms of utilizing the new hardware or software. It is only an interim solution which does little but delay the eventual necessary conversion[Oliver, 1976]. If the replacement option is selected, the original software is typically discarded in its entirety[Wolberg, 1983]. Replacement may be accomplished by purchasing or leasing a standard software package or by a new development effort. Since the termination of a system is highly unlikely and since emulation and replacement are typically inappropriate, conversion is the most frequently chosen alternative.

Although the GAO found the typical life cycle of computer hardware systems within the government to be seven years (architecture life cycles are longer), many existing programs in federal agencies were designed for, and are operating on, computer systems that are fifteen years old and older, before on-line systems, random access and telecommunications were generally available[Collica et al.,

1980]. Many of these programs use techniques common to second generation computers, such as tape-oriented batch systems, programming in assembly language or in very early versions of COBOL, FORTRAN or even in AUTOCODER.

In many cases, these programs perform vital functions, such as payments to retirees or agency personnel. Converting these programs is difficult, costly and time consuming so conversion is justified only if the hardware or software has become obsolete and is no longer supported by its manufacturer or if the system has reached saturation, i.e., there are not enough hours in a day to run all the required programs[Collica et al., 1980].

Once the decision has been made to convert software, there are basically three approaches or techniques which can be used in the process[Wolberg, 1983; Collica et al., 1980; Oliver, 1978]: recoding, reprogramming and redesign. These techniques are discussed in the next chapter.

Current Research Interest in Conversion

The Department of Defense Software Technology for Adaptable, Reliable Systems(STARS) Program Strategy expressed the concern of the computer community for more aggressive research in all areas of software engineering[Department of Defense, 1983]. Two items of interest which were defined as functional task areas are measurements and human engineering. This thesis research makes a valuable impact on this new initiative since software conversion measurements, including programmer productivity, are investigated.

Collica et al. listed several critical areas in need of immediate attention by researchers[Collica et al., 1980]. Two of these critical areas which were studied in this research are:

1. Conversion cost estimation guidance is needed.
2. The different types of people who are required for various types of conversions need to be identified.

Dunham and Kruesi emphasized the environmental specificity of resource estimating as an important issue[Dunham and Kruesi, 1983]. They pointed out that the accuracy of models typically vary considerably across different organizational environments. As one gains greater insight into the various environments by specific studies, the subject of effort estimation will become clearer.

Chrysler proposed programming research environments that require investigations such as he conducted to determine significant characteristics affecting development time[Chrysler, 1978]. He proposed no conversion environments, so this research adds a new dimension to Chrysler's suggestions for research.

Since there is currently no pool of professionals experienced in software conversion except for the few vendor firms that specialize in this area, an agency called the Federal Conversion Support Center(FCSC) was established in 1980 to assist federal organizations in planning and performing software conversions. Wolberg remarked that general software management planning procedures suggest the making of estimates of effort, resource requirements and project duration while considering the level of performance and productivity

of personnel[Wolberg, 1983]. There is a definite lack of research in the area of software conversion that addresses these management concerns. This research is just one of the many efforts required to fill the void that exists.

Air Force Base Level Data Automation Program

A current Air Force computer replacement effort, the Base Level Data Automation Program(officially titled Phase IV), is replacing over 225 obsolete base level computer configurations with about 150 new computers. It appears that the short title of Phase IV was chosen to indicate the fourth contractual agreement for Air Force base level support. All existing base level software systems are being converted to the new hardware. Several Air Force organizations are converting their own systems totaling about 3 million lines of code while a contractor is converting common systems which total about half that amount.

The Air Force Automated Systems Project Office(AFASPO) has primary responsibility for managing the Phase IV Program including the establishment of reporting requirements applicable to all participants. Since the inception of the program, the AFASPO has been receiving monthly status reports from all organizations involved in the conversion. In June 1983, the AFASPO requested that all organizations submit programmer resumes for all programmers participating in the conversion and basic program description or information forms for each program converted[AFASPO, 1983]. A

fundamental basis for this data collection was the Air Force's acknowledgement of software conversion as a significant area requiring increased attention.

The AFASPO plans to provide the raw data to the Rome Air Development Center(RADC) or some other research agency for its analysis upon completion of the entire conversion effort. The preliminary set of data was made available for use in this thesis research in order to perform an initial exploratory analysis. When the conversion effort is completed in late 1985, the total data can be analyzed using the procedures developed by this research.

Overview of Research and Chapter Summary

During the preparatory stages of this research, an extensive literature review was conducted. Chapter 2 presents an overview of significant literature related to the general area of software conversion. This chapter provides the framework for all that is to follow. The two major items of interest, programmer productivity and effort estimation, are discussed and reviewed in Chapter 3. Some of the literature in this chapter refers to software development since there are items of interest that overlap with software conversion.

An integral part of this research is the Air Force Phase IV software conversion effort. Chapter 4 provides the basic details of the Phase IV program which are essential for a complete understanding of the data collected.

The preliminary stage of the research involved a review of the data collection forms and the development of an encoding scheme for efficient data entry and manipulation. This stage included a basic and separate analysis of the resulting programmer resume file and the program information file. The first major step of the research required the merging of the program information file with the programmer attribute file to form composite records for analysis. The objective was to study the impact of programmer attributes on productivity. Chapter 5 presents the analysis conducted and the results obtained from investigating the sample data.

The last step of the research involved an evaluation of the accuracy of notable conversion effort/cost estimation models. Phase IV system level effort data was obtained to conduct this study and to permit the development of an effort estimation model specific to the environment. Chapter 6 details this work and the modifications of existing models performed in an attempt to improve their accuracy.

Finally, Chapter 7 presents a summary of the methodology formulated for use in this research plus a summary of the case study findings and conclusions. A discussion of future research possibilities is also included.

CHAPTER II

PANORAMIC VIEW OF SOFTWARE CONVERSION

The following definition of software conversion has been used in this research:

Software conversion is a process of transporting a program or system solely for the purpose of enabling such a program or system to execute correctly in an environment different from the one for which originally developed.

This chapter reviews the literature in the general area of software conversion while the following chapter presents the related cost/effort estimation and productivity literature.

Conversion Types and Techniques

A summary of the possible types of conversions and an assessment of the relative difficulty of the effort involved is presented in Table 1. Classifying conversions by source and target environments gives an indication of their wide variance as well as the level of difficulty which should be anticipated in different types of conversions. Within each environment, one can specify the same or different computers and/or languages. A and B stand for different computer hardware, L1 and L2 represent different languages and VX, VY, and VZ are different versions of a language for possibly different operating systems.

Table 1. Classification of Conversion Efforts

| Class | Source | | | Target | | | Difficulty |
|-------|----------|-------|------|----------|-------|------|------------------|
| | Computer | Lang. | Ver. | Computer | Lang. | Ver. | |
| 1 | A | L1 | VX | B | L1 | VX | Average |
| 2 | A | L1 | VX | B | L1 | VY | Difficult |
| 3 | A | L1 | VX | B | L2 | VZ | Most Difficult |
| 4 | A | L1 | VX | A | L1 | VY | Easy |
| 5 | A | L1 | VX | A | L2 | VZ | Highly Difficult |

Adapted from [Wolberg, 1983].

The conversion of existing software into another language involves a change of implementation language. These are shown in classes 3 and 5 of Table 1 where L1 and L2 are different. Notice that these two classes are of the greatest relative difficulty. Class 2 indicates a conversion from one machine to another using a new version of the existing language. This class is just below the level of difficulty of classes 3 and 5. Most of the software being converted in the Air Force Phase IV project is in class 2 while some is in class 3.

When a decision is made to convert the original software, there are basically four strategies, according to Wolberg's definitions, for completing the task[Wolberg, 1983]:

1. Translation -- The primarily automatic conversion of software.
2. Recoding -- The manual conversion of software.
3. Reprogramming -- The conversion process which includes some system redesign but no significant functional redesign.

4. Redesign -- The conversion option which includes a functional redesign of the system thus implying a level of software development.

Wolberg further states that translation and recoding use the original software as the specification for the new system. Reprogramming uses the original software plus the functional specifications to develop the new software. This new software will differ, to a varying degree, from the original software. Redesign is more expensive than reprogramming and will produce software that bears little resemblance to the original. He complicates the subject by defining the term "conversion" to mean a process with an important degree of translation and/or recoding. Thus, he makes a distinction between systems that are converted and those that are reprogrammed or redesigned. Oliver and Collica et al. present similar breakdowns and descriptions of conversion techniques; however, there are some differences that cause confusion when comparing all the descriptions[Oliver, 1978; Collica et al., 1980].

To clarify the techniques, there is a need to provide the basic list of software development documents upon which the software conversion is based. The document list, providing the framework for the conversion, is assumed to include:

1. functional specifications,
2. system design specifications, and
3. program descriptions(most importantly, the source code).

Each level of documentation obviously provides greater detail and more structure.

With this framework in mind, one can then differentiate between the conversion techniques by specifying the representation or documentation of the existing software used in the conversion. For this research, recoding is assumed to include a combination of automatic translation and manual visual inspection of the code, while reprogramming and redesign assume their normal definitions. In recoding, the existing source code is used as the basis while in reprogramming the system design specification serves as the basis of the conversion. In redesign, the functional specifications of the system are used in the conversion so that the user's viewpoint of the function of the software remains unchanged. A software conversion effort may involve only one of the recode, reprogram and redesign approaches or it may follow any combination of the three[Fernandez and Sheppard, 1984].

There are tradeoffs associated with these three approaches. Recoding is the easiest to do since each line of existing code is translated to an equivalent line(s) of code in the new environment. This translation can often be at least partially automated by writing a program or using a vendor's product to perform the line-by-line conversions.

Redesign is the most difficult approach to conversion since a new design specification is required before the programming can be done. Different algorithms, logic and program structures may be

used. Rarely can this type of conversion utilize automatic conversion aids. However, redesign does allow the maximum improvement in the system in terms of taking advantage of features of the new language or environment as well as any recent developments in algorithms.

Reprogramming ranks in difficulty between recoding and redesign. It involves an analysis of the system being converted based on the existing design documents. The same functions and algorithms remain but some new code with different logic may be included. It is significant that all three techniques provide the user with functionally equivalent software. This is the essence of the conversion process.

Management must take great care in deciding which technique(s) should be used for the conversion project. Collica et al. suggest the following criteria for this decision[Collica et al., 1980]. Recoding may be selected when the source and target languages are similar and the hardware/software capabilities of the source and target computers are comparable. Reprogramming is applicable when the source and target languages are dissimilar as when converting from a low level to a high level language. Redesign is the correct choice when the source program is many generations old, is poorly structured and documented and the design is out of date.

A type of software conversion that provides a unique set of difficulties is one that involves a data base management system(DBMS) environment. A conversion that involves COBOL programs utilizing a

DBMS can be as much as ten times more costly as the same programs in a file environment[Collica et al., 1980]. General solutions to DBMS conversion problems are still in the research stages primarily because there are many DBMS's in the marketplace but little commonality exists among them.

Fry et al. discussed the conversion of data base systems, detailing the problems involved and the automated tools undergoing research that may be useful for converting data bases and related programs[Fry et al., 1978]. These conversions are difficult because of the proliferation of data models and levels and styles of DBMS interfaces, internal data representations, and hardware architectures. Little work has been done in the area of DBMS applications because of their complexity. Shneiderman and Thomas described an automatic data base system conversion facility which provides one approach to coping with the data base conversion problems[Shneiderman and Thomas, 1982].

Oliver discussed several aspects of conversion which may involve technical difficulties no matter what type of conversion is involved[Oliver, 1978]:

1. Problems arise when the source machine sets special indicators or switches, such as, overflow, invalid data, etc., and the target machine either does not or does so under slightly different conditions.
2. The format and the amount of information that is specified to define a file varies among languages.

3. There is no "standard" format for the recording of variable length records on tape or disk.
4. File organization becomes a significant consideration on any nonsequential file since the processing of such files may vary by language and machine.
5. Data may be represented in different ways on the source and target machines.
6. Differences created by individual organizational programming practices can create significant problems.

Conversion Process

As is the case with new software development, there is no clear consensus on the definition of the steps or phases of a software conversion process. Table 2 presents three different views of the phases of software conversion. The three stages or phases of the conversion process as defined for Phase IV are used in this research. The first stage, the pre-conversion stage, includes a preliminary study, planning and data preparation. The conversion or second stage involves the actual conversion of the software and applicable testing. The third or post-conversion stage includes updating documentation, implementation, and the application of critical changes.

The preliminary study conducted during the pre-conversion stage is a very important step of the conversion process. It includes investigating all possible alternatives to insure that conversion is

Table 2. Comparison of Phases of Software Conversion

| Collica et al.* | Wolberg* | Air Force Phase IV* |
|------------------|------------------|---------------------|
| Planning | Planning | Pre-Conversion |
| Data Preparation | Data Preparation | Conversion |
| Translation | Conversion | Post-Conversion |
| Unit Testing | Testing | |
| System Testing | Implementation | |
| Parallel Testing | | |

*[Collica et al., 1980; Wolberg, 1983; Air Force Automated Systems Project Office, 1982b].

the best option available to obtain the required computing capability. The planning function will be discussed in a subsequent section of this chapter. The data preparation step is significant since there is typically a large volume of data that enters the conversion process. The preparation of test files that will execute a high percentage of the code is an important aspect of the pre-conversion stage. The FCSC provides some forms that can assist with this phase[Federal Conversion Support Center, 1982b].

The conversion stage involves using one or more of the techniques discussed earlier and testing the results. Substantial effort is required to correct logic errors that may have existed in the original source software or that were introduced by the conversion process. Generally, conversion problems are discovered when test data is passed through the software. Problems may continue to appear as additional integration occurs in system testing.

The post-conversion stage is the "bread and butter" stage for future conversions and maintenance of converted software. All the documentation is changed to reflect the new hardware and/or software environment. Even though few changes may be required in some cases, their significance can not be minimized. Operations in the new environment can be implemented in parallel with the old environment with changes being made to the software only after cutover of an application is achieved.

Lynn et al. discussed some management policies used to assist with the successful completion of a conversion effort[Lynn et al., 1979]. The overall management policy for a system which was undergoing conversion was to restrain new development as much as possible until the conversion was finished. Only the most essential changes to correct critical errors were permitted for programs undergoing conversions.

Conversion Planning

Experience in software conversion projects suggests that adequate planning and preparation is the key to success[Fernandez and Sheppard, 1984]. There are usually several steps included in the planning function. Wolberg's planning steps can be described as[Wolberg, 1983]:

1. Requirements analysis -- The first step is to prepare an inventory of the programs and files to be converted. The inventory should be updated as details, such as number of

statements in various languages, total number and types of records, etc., are determined.

2. Conversion guide preparation -- The guide identifies the differences between the source and target environments and covers all aspects of the conversion, including the conversion of programs, data, and so on.
3. Conversion methods determination -- Based on the previous two steps, the aspects of the conversion which will be performed manually and those which will be performed using conversion tools and aids are determined.
4. Estimation of required resources -- Estimation techniques are selected and applied. Wolberg specifically mentions that estimating productivity is a crucial step in the planning of a conversion project since it directly impacts the effort estimation.
5. Scheduling -- A schedule based on an estimate of required resources is prepared. The scheduling takes into consideration such obvious factors as hardware delivery dates.

If the conversion effort is to include much reprogramming and redesign, a functional analysis of existing systems could identify frequently used functions that might be developed into reusable modules[Fernandez and Sheppard, 1984]. The relative inefficiency of the converted code compared to the original software must be considered in the planning function. For applications where

degradation in space and/or time requirements cannot be tolerated, further attention from the programming staff will be required. Appropriate allowances for this effort should be included in planning for the conversion.

There are unique requirements that must be considered during the planning function. The training of programmers to write in a new language of the target environment must be planned for well in advance because of conflicts that may arise in current work schedules and schedules of available training classes. If the new language is Ada* then Ada training must even take programmers into an advanced "language mind-set" to produce proficient programmers[Fernandez and Sheppard, 1984].

An FCSC survey revealed that a major complaint echoed by all interview participants was that the time allowed was insufficient to adequately plan the conversion effort[Federal Conversion Support Center, 1983b]. A complex conversion may require as much as 40% of the effort devoted to planning, quality assurance and configuration management. It is important to realize that as unexpected variables become evident during the conversion, the plans will require revision. Management quite often underestimated the cost of conversion and the effort involved and therefore developed unworkable and unrealistic conversion plans. Better estimating techniques are needed to assist managers realize the scope of the effort and thus do better planning. The FCSC has published other documents specifically

* Ada is a registered trademark of the Department of Defense.

designed to assist the conversion planner in the preparation of plans and work packages[Federal Conversion Support Center, 1982b; 1983a].

Management Concerns

Wolberg stated that an important, but not well known, fact about conversions is that they tend to have more problems with management than technical aspects. One problem is management of the data involved in the conversion process. A conversion contractor, Rand Information Systems(RIS), has found that taking a single program from the source environment to the target environment may require, on the average, five or six data sets which include the source material, a couple of input files, a couple of output files and perhaps the master file[Wolberg, 1983]. This means that to convert 1000 programs, 6000 data sets may have to be managed. Critical to an organization's ability to manage the conversion is the control of materials, such as, record definitions, record layouts, system flowcharts and documentation on how the program is used.

Oliver related that there is a significant difference in emphasis between the management of a conversion project and the management of a development project. A conversion project requires, and allows for, more discipline and stricter adherence to procedures. A conversion may very well be an assembly-line type of operation, where the total effort is broken down into well-defined tasks which are more dependent upon experience and strict adherence to procedures than on innovation and ingenuity for their successful completion.

Oliver also pointed out that many of the ground rules for software production do not apply to conversions. An example is that manpower and time are not generally interchangeable in a software development project, but within certain bounds, they are in a conversion project[Oliver, 1979b].

Boehm observed that poor management can increase software costs more rapidly than any other factor[Boehm, 1981]. He made this statement primarily for software development projects but from Oliver's statements above one can conclude that this is even more true for software conversion projects.

Collica et al. were emphatic in their statement that managers are often threatened by conversions because they have neither planned nor budgeted for conversion[Collica et al., 1980]. Managers are eventually forced to convert in a timely manner with as little disruption to the ongoing system as possible. Managers must also deal with programmers who view conversion with equal disdain. Conversion programmers are called on to work with programs they neither designed nor coded on a machine with which they are not familiar to perform functions that are somewhat mechanical and not as intellectually stimulating as new coding or design. Managers must be prepared for this "people problem" challenge to avoid multiplying the difficulties involved. Getting programmers involved in the early stages of conversion, especially in the study and selection of conversion techniques, can make them feel more a part of the total project.

Chapin discussed the fact that management decisions play a critical role in determining the level of staff productivity in the maintenance of computer programs and systems[Chapin, 1981]. Historically, the builders of the programs and systems have acted as though someone else - the user and the maintainer - were going to pay the costs of use and maintenance. Chapin's observations apply completely to conversions, requiring managers to be concerned with the future impacts of current decisions.

The GAO suggested ways to improve software conversions and reduce their cost[General Accounting Office, 1977]. One suggestion was greater emphasis on quality in the original development of software and documentation. The possible greater cost of software development can be more than offset by easier and less costly future conversions. More widespread use of automated programmer productivity aids can also ease software conversion as well as software development problems. Conversion cost can also be reduced by management recognizing that most software will eventually be converted to new equipment and then taking steps to avoid the use of vendor-unique features. Because of the general lack of conversion expertise, Oliver stated that it would generally be wise for organizations to avail themselves of contractor support for conversion because of their extensive conversion experience[Oliver, 1978]. However, great care must be taken in pursuing the contracting of the conversion process.

The Contracting Option

A National Bureau of Standards(NBS) study pointed out that it is unlikely that many of the programming staff will have participated in a previous conversion at the same agency since the average time between computer replacement in federal agencies was found to be seven years, which is about double the tenure of a programmer/analyst at a given installation[Skall, 1982]. Many people on the conversion staff may also lack intimate knowledge of the software to be converted due to personnel turnover.

Much of the anxiety and uncertainty surrounding a conversion may be alleviated by contracting an experienced vendor. Contracting may be the only option available since hiring freezes and personnel ceilings prohibit many agencies from acquiring additional personnel for a conversion project.

The contracting option will not solve all of a manager's problems. Contracting may multiply the complexity of the entire effort so it must be approached with caution. The FCSC provides a document to assist with the contracting option[Federal Conversion Support Center, 1982b].

Conversion Tools

As in a program development effort, the size of the project determines the significance of using conversion aids. The workhorse of the conversion project is the automatic converter. There are numerous products on the market to convert one language to another

language and/or from one computer to another. Suppliers of suitable tools for the conversion effort may be found in the current literature. Datapro presents extensive conversion products in the Programming Aids section of their annually updated reports [Datapro, 1983]. The FCSC also published a list of software available with detailed information regarding function, applicable hardware and operating systems, source and target details, plus references to additional information [Federal Conversion Support Center, 1982c]. All the data in the report was verified by the conversion product vendors prior to its publication. The FCSC intends for the survey to be one of the primary sources of information on conversion tools available to federal agencies. This survey is upgraded annually by the FCSC.

The limitations of software conversion tools must be realized. Collica et al. discussed the problems of the semantic definitions of source code [Collica et al., 1980]. They related that automated tools typically solve the easy problems which are encountered in conversion, such as the differences in syntax between two languages. However, the semantics of a block of code can sometimes be determined only by interfacing directly with the author(s) of the code. Automated translators usually will not translate the correct semantics of a block of code if the semantics cannot be identified by merely scanning the code.

Collica et al. provided an example of problem code for a translator [Collica et al., 1980]. The following block of FORTRAN

code written for a CDC 6700 computer which stores 10 characters per word is to be translated into IBM FORTRAN to run on an IBM 360/370:

```
COMMON A(4), B(4)
DO 100 I=1,4
  B(I)=A(I)
100 CONTINUE
```

Since these lines of code are syntactically correct IBM FORTRAN statements, a translator would probably not modify them. However, the arrays A and B may have been used to store a 40 character string on the CDC machine with 10 characters per word. Since the IBM machine stores 4 characters per word, the code should be modified to:

```
COMMON A(10), B(10)
DO 100 I=1,10
  B(I)=A(I)
100 CONTINUE
```

A translator would probably not be able to "know" if array dimensions and/or loop counters for a block of code were set up to handle character data. Tools are not an end in themselves, but rather one part of a complex management strategy required for the conversion process.

CHAPTER III

CONVERSION EFFORT ESTIMATION AND PRODUCTIVITY

Estimation and productivity involve measures of one type or another and the importance of studying them is made clear from Boehm's quote of Lord Kelvin[Boehm, 1981]:

When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science.

Mills stated that there are two parts to an estimate: making a good estimate and making the estimate good[Mills, 1980]. Effective management and software engineering techniques must surround both parts of Mills' estimate. In addition, part one, making a good estimate, is directly concerned with the use of cost estimation models. Part two of Mills' definition includes a concern for productivity.

There exists a direct relationship between cost/effort estimation and productivity. High productivity implies a faster conversion and thus low effort measurement while low productivity implies a slower conversion and high effort measurement. This thesis research separated the two measures to analyze each individually and then synthesized the results. This chapter divides the literature review into two sections, one for effort estimation and the other for productivity.

Little research with software conversion effort estimation and software conversion productivity has been reported in the literature. Therefore, effort estimation and productivity research for new software development was included when it contributed in some way to the study of the parallel topics in software conversion.

Cost/Effort Estimation

The definition of software engineering economics provides a framework for the study of conversion effort/cost estimation[Fernandez, 1982]:

The application of mathematical and managerial techniques to the design and development of software products to derive good cost and schedule estimates which compare favorably with the actuals.

To make this definition more applicable, one can change the term "development" to "development or conversion." Good effort/cost estimates are critically important issues of software conversion.

Estimation is typically preceded by the term "cost"; however, authors are often quick to point out that they are really discussing "effort" estimation and that an "effort" estimate becomes a "cost" estimate by multiplying it times the average manpower cost of the organization. The ensuing discussion will use the terms "cost" and "effort" interchangeably with "effort" being the preferred term. If costs are being specifically addressed then "cost estimation" is the more appropriate term.

Software development models typically yield estimates that are very different from those of conversion effort models. Wolberg presented a comparison of a conversion effort prediction model and two software development models revealing that conversions require much less time than development projects[Wolberg, 1981]. This discussion is primarily oriented to recoding since Wolberg considers redesign projects to be similar to new development projects and reprogramming projects to require approximately half the effort of new development[Wolberg, 1983]. A conversion study may review software development estimation models to search for directions for analysis but not to select an applicable model.

Basic Estimating Concepts

Effort estimating techniques can be divided into three categories or methods[Federal Conversion Support Center, 1981a]:

1. Experience Method.
2. Task Analysis.
3. Parametric.

The experience method is primarily based on expert judgement while task analysis pertains to a method which utilizes a handbook approach to estimating. The parametric method refers to the use of single or multiple variable equations in estimation. The models studied and developed in this research are of the parametric type.

Dunham and Kruesi divide effort estimating models into two basic categories: total cost and cost distribution[Dunham and Kruesi,

1983]. Total cost models are simply those of parametric type. Cost distribution models differ with total cost models in that they focus on resource expenditures over time. Putnam's model is an example of such a type as it predicts time-dependent parameters[Putnam, 1980]. With conversion times a great deal shorter than those for software development, total cost or parametric models are more appropriate for this research.

When developing a parametric model, it is important to recognize the influence of the organizational environment[DeMarco, 1982; Wolberg, 1983]. Published models typically only provide an initial "ball park" estimate that must be adjusted for each organization[Wolberg, 1983]. The organizational sensitivity of estimation may be lessened by developing models based on organizational data.

Parametric models generally use lines of code as one of the major parameters, if not the only parameter. DeMarco related that the most serious objection to using lines of code in a model is that one cannot count lines of code at the beginning of a development project; the count has to be estimated[DeMarco, 1982]. This objection does not apply to conversion estimation models since a count of lines of code is always available at the start of the project.

Oliver, a proponent of the task analysis method, criticized parametric models as worse than useless[Oliver, 1978]. It is obvious that Oliver has had some bad experiences which influenced his

comments. This thesis research studied some effort estimation parametric models which were found to be quite adequate for the Phase IV environment.

Conversion Effort/Cost Estimation Models

The FCSC conducted an extensive study of conversion cost/effort estimating techniques and models[Federal Conversion Support Center, 1981a]. All were found to be inadequate for one reason or another. Therefore, the FCSC developed a parametric model with a foundation of a work breakdown structure incorporating some of the better elements of existing models. The initial version of the FCSC model was formally called FCSC Hybrid Conversion Cost Model. After some experience with the model, another version was developed and documented in a subsequent report[Federal Conversion Support Center, 1982a].

The FCSC model provides estimating methods, formulas or guidelines for the following conversion tasks:

1. Conversion planning and analysis.
2. Work package identification and preparation.
3. Test data generation.
4. Application program and system software conversion.
5. Data file and data base conversion.
6. Operation control language(ACL) conversion.
7. System testing.
8. Redocumentation.
9. Acceptance testing.

10. Conversion management and administrative overhead and/or contract administration and support.

11. Miscellaneous resource estimating and costing.

The details of the estimation procedures for each of the tasks are included in Appendix A. Though the entire model is presented here, only those tasks that parallel those used in the Air Force Phase IV program will actually be included in this research. Basically, the tasks of primary interest are those numbered 3 through 9 above.

The Hahn and Stone model appears to be the earliest attempt to develop a conversion parametric effort/cost estimation method[Hahn and Stone, 1970]. The model defines three cost categories of interest:

1. Cost of transferring programs.
2. Cost of transferring data.
3. Other costs.

The major portion of the model deals with the cost of transferring programs with the main emphasis being placed on the costs of manual conversion or recoding. The details of the model are provided in Appendix A.

Grim, Epler and Andrus studied Air Force conversion cost estimation and found no comprehensive method available[Grim, Epler and Andrus, 1978]. They developed a model which addressed the following costs:

1. Application program conversion.
2. Data conversion.
3. Operating procedures conversion.
4. Other costs.

The cost of converting application programs was subdivided into costs for analysis, programming, manpower and machine use. The programming cost estimation method received the most attention. The details of the Grim, Epler and Andrus method are included in Appendix A.

The AFASPO suggested a slightly modified version of the Hahn and Stone model for use in the Phase IV program[Air Force Automated Systems Project Office, 1982a]. Only the software conversion cost estimation guidelines were included. Also, the details of the model, found in Appendix A, reveal that only COBOL to COBOL conversions are addressed since the majority of Phase IV conversions are of this type.

Wolberg analyzed nine Rand Information Systems(RIS) conversion projects and developed a regression model for estimating effort based on lines of code[Wolberg, 1983]. Though not stated by Wolberg, the model was actually based on only the seven projects which were high level to high level language conversions. This became evident when the RIS data was subjected to regression analysis to determine the R^2 (coefficient of determination), which was approximately 0.61. In addition to a recoding estimation model, Wolberg also suggested models for reprogramming and redesign. It should be noted that the RIS data included total project times, from planning to implementation. Details of Wolberg's models and methods are included in Appendix A.

Development Models of Interest .

Boehm developed some cost estimation models for new development efforts which he extended for use in conversion projects[Boehm, 1981]. Boehm suggested using his models for conversion by applying specific cost driver factors and considering conversion as an instance of adaptation of existing software for a new application. He recognized the limitations of this approach when he encouraged the reader to review the work of the FCSC for cost estimation models.

Najberg was commissioned by the Air Force Electronics Systems Division(ESD) to perform an analysis of the resource and schedule estimates for the Air Force Phase IV Program[Najberg, 1981]. He obtained estimates, mostly of expert judgement type, from the organizations(MAJCOMs) to be involved in Phase IV. Using ESD's Standard Parametric Software Cost Estimation Model(SPSCEM), Najberg calculated conversion effort estimates and compared them to those submitted by the MAJCOMs. The results showed that there were great differences between the two estimates. The SPSCEM calculates estimates using the average of four software development effort estimation models. Najberg stated in a private interview that the SPSCEM model is being completely revised since the results obtained were not acceptable[Najberg, 1983].

Basili and Freburger also performed an analysis of 15 software projects developed for NASA/Goddard Space Flight Center by Computer Sciences Corporation[Basili and Freburger, 1981]. Though the projects were all new development efforts, the concept of reused code

used in the study might have some applicability to conversion efforts. One of the equations, derived from the data, used the number of developed lines of code(DL) as the predictive variable for man-months of development effort(E). DL is a derived quantity equal to the number of new lines of code(total lines minus reused lines) plus 20% of the reused lines(all line counts are in thousands):

$$DL = NL + 20\%(L - NL)$$

where

NL = new LOC; that is, not reused code

L = total LOC

L-NL = reused code.

This predictor variable(DL) was used by Basili and Freburger to derive an equation for effort estimation:

$$E = 1.48 * DL^{0.98}$$

The research conducted by Chrysler was a unique study of development time as affected by processing characteristics of programs and experience characteristics of programmers[Chrysler,1978]. Chrysler developed a regression equation with only five of the 60 variables hypothesized to have an impact. The utility of the predictive equation is limited somewhat by the details of the program that must be estimated for use of the equation. The result of major interest is really the fact that a researcher has studied the significance of both programmer and program characteristics in estimation or prediction of programming time.

Jeffery and Lawrence developed equations for predicting programming time(coding and unit testing) based on the data from three different organizations[Jeffery and Lawrence, 1979]. The independent variable used was procedure(COBOL) lines of code and the equation coefficients were significantly different for all three organizations.

Software Conversion Productivity

There are several studies in the literature which relate to productivity though none of these is in the area of software conversion. The general concepts, definitions, and measurements involving productivity are applicable to software conversion; therefore, significant work in this area is discussed.

Productivity Measurements

There is some debate in the literature over the selection of an appropriate measure of program development or programmer productivity. This is exemplified by Johnson's paper in which he related that the measure of lines of code is the "only usable measure" of development productivity[Johnson, 1977]. He also expounded the concept that productivity is only productivity if it is measurable. There are variances in how LOC are measured and reported so understanding the underlying definitions is fundamental. Precise definitions of programs, man-days and LOC are necessary to avoid difficulties in reporting and comparing productivity.

Johnson used Brooks' categories of programs to explain the inherent differences in the levels of complexity of programs under development[Brooks, 1975]. It is important to define one's level of program complexity when discussing productivity rates. A distinction must be made between productive and nonproductive time. Also requiring clarification is whether design time is included in computing LOC rates. If these rates are based only on programming/testing time, the result could be twice the rate that would result if all phases of the project were included. The last definition requiring clarification is that of LOC, since there are various ways of counting source statements. LOC could be the total of all source statements, comments and job control statements, COBOL Procedure Division statements or various combinations.

Jones presented the unit of cost of programming as an alternate measure of productivity[Jones, 1978]. He discussed two measures of programming productivity: speed and cost. The units of measure of programming speed he called work units since they related to the speed at which a programmer works. The primary example of a work unit is the typical measure of lines of code written per programmer-month. Cost units are units of measure of programming cost and a significant cost unit is that of programmer-months of effort per thousand lines of code.

Jones related some problem areas in using lines of code per programmer-month; e.g., sensitivity to line-counting variations, ineffectiveness for noncoding tasks and attention focusing on the act

of coding of a program, which is a misdirection, since this is but a small part of the total effort required. It appears to this researcher that cost units will still be sensitive to line counting variations and that the calculations of cost units for noncoding tasks add a degree of confusion to the process. In addition, the use of work units, as called by Jones, are still exceedingly popular even in cases where reused code plays a great part.

Crossman presents another alternative to measuring programmer productivity[Crossman, 1979]. He encountered numerous difficulties in attempting to measure programming productivity especially because of the many subjective assessments that must be made in the process. It was decided to try to measure the number of functions within a program, with function defined as a section of the program that performs only one activity, conforms to the permitted logic structures of structured programming including one entry and one exit point and has about 5 to 50 source statements. It also appears that the number of paragraph names in the program is a very good estimate of the number of functions.

A problem with this entire effort is the variations in the definitions of functions. Also, so much work has been done with LOC that it may be useful to take the number of functions calculated and derive a figure for LOC that could then be used as appropriate. Further research with the concept of functions seems necessary.

Productivity Studies

The productivity measurement and estimation study of Walston and Felix is one that is continually cited in the literature [Walston and Felix, 1977]. They mention that their research was geared towards measuring the overall productivity of projects and not that of individual project members. The definition of LOC is the count of source records input to a language processor including job control language, data definitions, link edit language and comment lines. Reused code is not included; however, no definition is provided for reused code to determine exactly what is excluded. Effort is measured in man-months required by the project, including management, administration, analysis, operational support, documentation, design, coding and testing.

The programming productivity measure used in the study was the ratio of delivered source LOC to the total effort in man-months. A set of 68 variables was selected and analyzed against the data base to determine which variables were significantly related to productivity and the result was that 29 variables, including programmer experience, showed a high correlation with productivity.

Chrylser expanded the study of productivity by hypothesizing fifteen program characteristics and five experience characteristics of programmers as being associated with increases in programming time [Chrylser, 1978]. The \log_{10} of programming hours for sample programs was regressed against the \log_{10} values of program characteristics and the following program characteristics which were found significant at the 0.05 level or less are:

- * output fields and output records
- * control breaks and totals
- * input fields, files and records
- * output files and report formats
- * mathematical operations
- * output fields without corresponding input fields

Again, the \log_{10} of programming hours was regressed against the \log_{10} of each programmer characteristic. Years of formal education and age were also used in the regression. All these characteristics were found to be significant at least at the .005 level:

- * age in months
- * total months of experience programming
- * months of experience with business applications
- * months of experience at this facility
- * months of experience programming with COBOL compiler in use at this facility
- * months of experience programming with COBOL compilers
- * years of formal education.

Chrysler's results indicated that each of the experience variables showed a significant relationship with programming hours. The age variable seemed to be a statistical repository for the cumulative impact of all the experience variables since it exhibited the strongest correlation with programming time probably because age showed a strong relationship with each of the experience variables. The formal education variable, though significant, is too general and it should take into account the type of education or curriculum.

Chrysler then consolidated the program and programmer characteristics, as discussed earlier, and produced a predictive equation for development time using the stepwise regression program. With only the following five variables in the equation, the multiple correlation coefficient was 0.836:

- * programmer experience at this facility
- * number of input files
- * number of control breaks and totals
- * number of input edits, and
- * number of input fields.

A parallel study, which dealt with programmer performance, investigated program and programmer factors which were significant [Schneider et al. 1981]. Two subpopulations of programmers appeared from the sample data:

- Novice: ≤ 4 computer science courses and < 3.0 GPA
or ≤ 2 years programming experience
- Expert: ≥ 7 computer science courses and ≥ 3.5 GPA
or ≥ 5 years programming experience.

A measure of performance was selected and the best predictor of the experts' performance was found to be the number of years of programming experience. For novices, the best predictor model included the number of computer science courses taken and computer science grade point average. The performance differences found between the two groups of programmers demonstrated that productivity is not only dependent upon program complexity but also on the

interaction that arises between program factors and programmer attributes.

Paulsen studied the relationship of productivity to program composition and program size[Paulsen, 1981]. Productivity was measured in terms of changed source instructions(CSI) produced in one person-year(PY) because most of the products developed by her organization, Santa Teresa Laboratory(STL), are modifications and/or enhancements of existing IBM products. CSI includes new as well as modified lines of code for the total product(TSI).

Paulsen's study of STL product development revealed that when productivity is plotted against the ratio(CSI/TSI) of changed source instructions(CSI) to the total number of statements(TSI), a convex curve results. When the CSI/TSI ratio is less than approximately 0.50(that is, the reused code is greater than 50 percent), productivity has a positive slope. When the CSI/TSI ratio is greater than approximately 0.50(that is, the reused code is less than 50 percent), productivity has a negative slope. The increasing slope of productivity when the percentage of reused code is high is due to the fixed overhead effort involved in handling a program. As CSI increases within this range of reused code, productivity(CSI/PY) will increase because more lines of code will be changed within the time expended which is dominated by fixed overhead effort. When the percentage of reused code is low and the number of CSI increases, the overall effort involved is no longer dominated by the initial fixed overhead effort. Therefore, within the range of a lower percentage

of reused code, productivity decreases as the number of CSI increases. Since compatible language conversions can be considered to have a high percentage of reused code(code translated automatically), it is not entirely unlikely that productivity will increase with lines of code. This was true of the Air Force data but it can not be concluded to apply to conversions in general. The overall result of Paulsen's study was that productivity was affected by program size and by the amount of reused code. No consideration was given to programmer attributes.

Basili and Freburger's study mentioned earlier found that productivity increased as the percentage of reused code increased[Basili and Freburger, 1981]. This is intuitively clear since the reuse of code should be less expensive than creating the code from scratch. In this study, productivity was measured in terms of the total number of delivered lines of code(expressed in thousands of lines) which included data definitions, comment lines and source statements which served as input to a language processor. The denominator of the productivity ratio was total effort which was defined as the total number of man-months of effort used on a project, starting when the requirements and specifications become final through acceptance testing. Effort includes programming, management and clerical time, such that, one man-month of effort is defined as 173.33 man-hours.

Basili and Freburger presented a productivity model they developed from their data base. They defined productivity as a function of the ratio of new lines of source code to total delivered

lines of code without any consideration of program complexity or programmer attributes:

$$P = 698 * RNTOL^{-0.75} \text{ where:}$$

RNTOL = ratio of new lines of code to total delivered lines
of code.

This model suggests that productivity is lowest when there is not reused code. Although Paulsen's data was from a very unique environment, Basili and Freburger's model results parallel those of Paulsen where the number of changed statements were over 50% of the total product [Paulsen, 1981; Basili and Freburger, 1981].

Lawrence conducted an expanded study of productivity as it related to the programming environment, programmer experience and programmer methodology [Lawrence, 1981]. From earlier evidence, Lawrence concluded that lines of code was a reasonable measure of output to use in a productivity metric. Productivity is defined as PL/T where PL is the number of procedure LOC and T is the total time in man-hours put into the job by the programmer from the receipt of program specifications to the completion of program testing. Comment lines were excluded from the count of PL.

The study concluded that there was no significant difference in the productivity of the four industry groups studied: semigovernment, banks and insurance, manufacturing and mining, and software houses. However, the results indicated that some organizations were obtaining significantly higher productivity than others. The results also showed that trainee programmers have a lower productivity than

intermediate and experienced programmers; however, no observable difference was seen between the intermediate and experienced groups.

The regression analyses revealed that the inclusion of both organization-identity variables and program and programmer variables provide a 50% increase in R^2 (coefficient of determination) compared with either set of variables taken by themselves. The results indicate that there are organization environment variables that, if identified and measured, could lead to a better regression equation for productivity.

Jeffery and Lawrence performed an inter-organizational study of programming productivity [Jeffery and Lawrence, 1979]. Two of the three organizations showed that productivity was moderately correlated with experience (measured by years on the job) while the third organization showed no such correlation. This counter-intuitive result was recognized by Jeffery and Lawrence who offered two possible explanations for the result: (1) after commercial programmers learn their craft, additional experience makes little difference, and (2) an inverse experience or skill relationship exists as the better programmers are promoted quickly to systems analysis or management, leaving the less skillful programmers as those with the greater number of years of experience. There is nothing in their sample data that provides a specific reason for this result. It is interesting to note that this study was the only one reviewed that exhibited this counter-intuitive result which indicates that a measure of experience, other than years on the job, may be necessary.

Programming time and program size were found to be highly correlated. It is significant that despite major differences between the organizations studied, the equations developed for programming time were very similar, both in terms of the model and the coefficients. Organization three exhibited a positive correlation between productivity and procedure LOC indicating that productivity rises as program size rises. This seemingly counter-intuitive result was perhaps due to the programming style at organization three which was observed to have a tendency to reproduce sections of code rather than packaging the code as modules and using call and perform statements. There is a clear parallel between the style of organization three and software conversion where large sections of programs are converted by an automatic translator. This supports the results of this thesis research which show that productivity rises with program size for the Phase IV conversion sample data.

Wolberg stated that estimation is a "tricky" procedure because of the many variables affecting productivity[Wolberg, 1983]. Using the effort equation he developed, Wolberg formulated a model for productivity(P) measured in lines per day. The effort equation was:

$$E = 7.14 * L^{0.47}$$

Assuming 22 working days per month, productivity is calculated as:

$$P = (L*1000)/(E*22)$$

which results in the following:

$$P = 6.37 * L^{0.53}$$

Wolberg reiterated that his effort equation is based on man-hour data that included all the staff time used in the RIS conversion projects. Therefore, his productivity measure should be treated as an estimate of total productivity with all project hours included; i.e., planning, data preparation, conversion, testing, implementation, documentation, etc. It is important to realize that to use productivity measured in programmer hours only, to develop a project budget, would be a grievous error.

Since many of the software development and software conversion cost estimation models include productivity estimates in their equations, valid productivity measures are a major concern. This thesis research is geared towards studying the factors that affect productivity rather than to determining another productivity measure. Examples are the statements by Oliver that knowledge of the application is not critical in performing the conversion and that the programmers of a system may be the worst qualified to convert it since they may not resist the temptation to "improve" the system while converting it[Oliver, 1978].

Jeffery and Lawrence provide a summary and discussion of the apparent inconsistencies of prior productivity research results and definitions[Jeffery and Lawrence, 1981]. All studies reviewed, except that of Jeffery and Lawrence, concluded that experience significantly influenced development time[Jeffery and Lawrence, 1979]. In this one case of Jeffery and Lawrence only one of three organizations showed no correlation between experience and

development time and they point out that this could be due to the rapid promotion of good programmers to levels of management which would produce a group of programmers whose years of experience might not reflect their level of skill. Johnson's study, which does not include reused code, revealed that productivity declined on the average as project size increased[Johnson, 1977]. On the other hand, Jeffery and Lawrence's and Basili and Freburger's studies, which included reused code, found that, on the average, productivity remained relatively constant as the size of the project changed[Jeffery and Lawrence, 1977; Basili and Freburger, 1981]. Paulsen's work showed that when the amount of reused code is approximately 50% or more, productivity has a positive slope but when the amount of reused code is between 0 to approximately 50%, productivity has a negative slope, thus productivity is depicted as a convex curve with the high point relatively close to 50% reused code[Paulsen, 1981]. This phenomenon explains why Walston and Felix's study of IBM systems, with much reused code, revealed that productivity increased as the percentage of reused code decreased[Walston and Felix, 1977]. This also indicates why Basili and Freburger's work, which included less than 50% reused code, showed productivity decreased as the percentage of reused code decreased[Basili and Freburger, 1981]. Paulsen stated that products with few new or modified LOC(small size) tended to have lower productivity because of what she called the fixed cost overhead. In the area of software conversion where the source and target languages

are very similar, the amount of "reused code" is close to 100% so it is logical to expect productivity to increase somewhat as the size of the program increases.

Differences also exist in the independent variables selected for the research. For studying programming productivity, the concern is for the time spent by the programmer and Jeffery and Lawrence included only the hours recorded from receipt of program specifications to delivery of tested code[Jeffery and Lawrence, 1979]. Walston and Felix included time from the inception of a project to its implementation while Basili and Fieburger included management and clerical overhead with programming effort. Wolberg's conversion effort study also included time from the administrative initiation of projects until final implementation[Wolberg, 1983].

The apparent lack of consensus in the literature is primarily due to differences in terminology, in the sample data, and in the specific variables being considered by the researchers. One element of new development research which requires strict definition to increase the commonality of the various studies is that of reused code. Also requiring clarity are the studies' definitions of LOC and programming time. Although there appear to be no apparent differences in the productivity of the industries studied by Lawrence, the results indicate that some organizations are achieving significantly higher productivity than others[Lawrence, 1981]. This thesis research shows that program size, programmer experience, program complexity and organization have a definite impact on the conversion productivity of programmers.

CHAPTER IV

AIR FORCE PHASE IV PROGRAM

Since the data for this research is being provided by the Air Force, it is fitting to include a detailed presentation of the Air Force's computer replacement program called the Base Level Data Automation Program and officially short-titled Phase IV(fourth contractual agreement for base level data automation support). This not only assists in the discussion of the general area of software conversion but also provides an analysis of the environment from which the software conversion sample data for this research is drawn.

Program Prescription

The Air Force Phase IV Data Project Plan provides an overview of the program since its inception[Air Force Automated Systems Project Office, 1982b]. The Phase IV Program was established in April 1976 to modernize existing base level computer hardware systems. The formal requirements specifications, Data Automation Requirements(DAR), for the Program were completed in September 1976 and submitted for certification. The Assistant Secretary of the Air Force for Financial Management(SAF/FM) granted the conceptual certification in October 1976.

In February 1977, the SAF/FM was designated the source selection authority(SSA) and in April 1977 the Air Force Automated Systems Project Office(AFASPO) was chartered to function as the Phase IV

Program Manager. The SAF/FM granted definition certification in March 1978 and the performance specifications for the request for proposal(RFP) were completed in April 1978. The General Services Administration granted Delegation of Procurement(DPA) to the Air Force in August 1978 and with the SAF/FM's approval the RFP was released in December 1978.

In April 1979, the GAO began a review of the Phase IV Program as requested by the Chairman, House Government Operations Committee(HGOC). Hearings were held before this Committee while RFP responses were being received and evaluated. In October 1979, the GAO released its report of findings which was critical of several elements of the Program and recommended that it be canceled. In November 1979, GSA suspended the DPA pending action by the HGOC. The Air Force immediately responded to the HGOC detailing significant weaknesses in the GAO's analysis and presenting a strong case for continuing the Program. However, the HGOC accepted the GAO's recommendation requesting that the Air Force consider establishing regional centers instead of doing an across the board replacement of all existing computers. In January 1980, GSA and the Air Force agreed on a redirection of the Phase IV Program.

The DPA was reinstated in April 1980, after the HGOC reviewed the Program redirection. Contract negotiations were resumed and in December 1980, Burroughs and Sperry-Univac Corporations were awarded contracts to competitively transition a set of software systems and demonstrate adequate performance on their own hardware. In October

1982, an operational test of the contractors' hardware and converted software was initiated. A study of the results lead to a decision in January 1983 to award the Phase IV contract to Sperry-Univac.

Phase IV Materialization

About 150 new computers(Sperry-Univac 1100/60) will replace over 225 base level computer configurations of Burroughs B3500/3700/4700 and Sperry-Univac U1050-II computers. The Burroughs machines support such functions as finance, procurement, personnel, etc. while the Univac 1050-II's support the base supply function. The hardware installation and software conversion and implementation is being accomplished in two increments. These increments are referred to as the X1 and X2 workloads. The X1 increment involves the replacement of the Univac 1050 systems with associated software, while the X2 increment involves the replacement of the Burroughs machines and associated software.

Sperry-Univac is responsible for converting about 20 standard software systems(1.5 million LOC) while about 25 Air Force development centers(primarily major air commands) are responsible for converting about 300 software systems(3 million LOC) that are unique.

One of the objectives of the Phase IV Program is to extend the data processing support of base level users through the 1990's by acquiring upgradeable/expandable hardware from a single vendor's family of equipment. Phase IV planners are expecting the useful life of the Sperry-Univac equipment to be 12.5 years from installation. A

12 year economic life was used for costing purposes with the remaining half year being used for the removal and disposition costs of the equipment. For life cycle budgeting, Phase IV permits two negotiated contract extensions with Sperry-Univac for a possible total of about 20 years under one ADP Program. The life cycle cost estimate for the Phase IV Program(in 1977 fiscal year constant dollars) is \$2.3 billion while the net impact on the Air Force budget is a life cycle cost decrease of about \$350 thousand indicating that Phase IV is actually going to save money.

There will be 63 single system bases, 28 dual system bases(56 computers) and 14 regional centers(27 computers). In addition, two X2 workload systems will be installed at the Defense Mapping Agency and there will be two transportable systems.

Conversion Assistance, Tools and Procedures

The Air Force Automated Systems Project Office(AFASPO), having primary responsibility for managing Phase IV, developed a guidance package to provide Air Force major air commands(MAJCOM) and Separate Operating Agencies(SOA) with information needed to effectively plan and accomplish their software conversion efforts[Air Force Automated Systems Office, 1982a]. An understanding of the guidance/information provided in this package permits one to appreciate the environment within which the conversion is taking place and uniquely defines the data used in this research.

The AFASPO guidance package called for the MAJCOM/SOA unique software to be converted as follows:

1. COBOL -- Automatic recoding, as defined in this dissertation, of Burroughs extended COBOL-68 to Sperry-Univac COBOL-74 will be accomplished with a Sperry-Univac automatic translator of 90% effectiveness rate(minus the ENTER verb). Manual recoding will be applied to complete the task.
2. FORTRAN -- No automatic translator was thought to be necessary because of the small number of FORTRAN programs so manual recoding is to be used.
3. Burroughs Assembly -- All systems written in Burroughs Assembly are to be converted by means of reprogramming, as defined in this dissertation.
4. AFOLDS -- Air Force Online Data System(AFOLDS) programs will require only minimal manual recoding since the contractor is to convert the AFOLDS such that it accepts currently existing programs.

Ten Sperry-Univac computer systems will be available throughout the country to support the conversion effort of the 29 MAJCOM/SOA development centers. Each center was assigned to a specific system for the conversion effort which would thus be started prior to the installation of the new equipment at most sites.

The contractor is also supplying utilities to convert data files from the existing machines to the Univac 1100/60. Also included are utilities necessary to validate the data files converted.

The Air Force is to provide to all development centers the following tools:

1. Percent Execute -- This system inserts probes in a COBOL program to monitor its execution. It is used to determine what parts are being executed by a given data set, thus permitting the development of more complete test data sets.
2. Documentation Aids -- The current language processors plus the COBOL and Assembly concordance tools provide variable usage and cross reference information. In addition a flow chart generator is provided for COBOL programs.
3. Phantom Paths -- This tool isolates "dead" code which cannot be reached and which can be deleted since it serves no useful purpose.
4. Burroughs Filter -- This tool is used to identify code which has a high probability of not translating to the new equipment and thus should be removed if possible.
5. Automated Compare -- Automatic comparison of files, as mentioned above, is provided.

Testing procedures require that the functional equivalence between the original and converted code be proven by means of visual verification of output.

The guidance package calls for preparatory or pre-conversion tasks, to include the following:

1. Removal of "dead" code. Phantom Paths may be used to facilitate the isolation of this code.

2. Improvement of code by replacing complex source code with simpler code and removing verbs which have a high probability of not converting automatically. The Burroughs Filter may be used.
3. Identification of problem code and removal if possible. Some segments of code which are in this category are: code sensitive to changes in collating sequence, code which uses Burroughs extensions or assembly language and code which relies on special operating system or hardware features.
4. Improvement of documentation. Documentation standards must be followed and a set of standard documents must be available to support the conversion of each system.

Resource estimation, a pre-conversion task, was to be accomplished using the Hahn and Stone model as modified by the Air Force or an Air Force devised average method[Hahn and Stone, 1970]. These are discussed in Appendix A.

Initial Conversion Experiences

Sperry-Univac converted an initial system of 26 COBOL-68 programs. This conversion was analyzed by the Air Force and this led to the AFASPO STC 404 report which includes much useful information for the remaining conversions[Air Force Automated Systems Project Office, Undated]. Some assumptions listed are the following:

1. The recoding of major command software will be strictly recoding with no improvements. Allowing modifications during conversion increases risks.

2. New capabilities offered by the target environment will not be exploited during the conversion. This is again to minimize risk.
3. The format, structure and medium of all input and output files will not be changed until the software has been transitioned and successfully implemented.
4. All source and data files on the Burroughs systems will be copied to tape using the File Management System(FMS). The Univac 1100/60 has the capability to process these FMS tapes.

It was discovered that the bulk of the existing documentation for Burroughs systems could effectively be transferred to the new environment. Most of the changes needed applied to the operators manual and the majority of these changes related to job control language and Burroughs terminology. Very few changes were necessary for the users manuals. All these changes can be highlighted before the conversion so that changes can be made easily.

The evaluation of this initial conversion process reveals that all programs increased in size after going through the translator and even more after the manual recoding of the source lines that were not automatically translated. The increase in program size is typical of what can be expected by conversion programmers. A 28,569 line system that was recoded by a contractor, using automatic and manual translation from B3500 COBOL-68 to Sperry-Univac COBOL-74 was analyzed. An increase of over 20% to 35,279 lines of code was discovered. The Air Force plans to improve the efficiency of these

converted programs only after the initial effort is completed and the converted programs compile and produce the required output. This approach is both less costly and less risky. Following this type of approach, if the resulting system operates as specified within the allowed time and space constraints nothing further needs to be done to the code.

The discussion above refers primarily to a contractor's early experience. This thesis research is concerned with an analysis of early conversion experience data of Air Force programmers. Subsequent chapters relate the data encoding and analysis performed along with the results obtained by this study.

Conversion Effort Data Collection

Since all of development or conversion centers are scattered throughout the country (two are outside the continental United States) and function somewhat independently, an automated method of collecting data was infeasible. Therefore, data was requested in raw form using AFASPO developed instruments or forms which were completed and forwarded to AFASPO. The basic requirement was for conversion centers to submit a monthly report to the AFASPO detailing the number of manhours expended for pre-conversion, conversion and post-conversion tasks[Air Force Automated Systems Project Office, 1982a]. The monthly report format was subsequently changed to one which requests a percentage of completion be provided for each system the center is converting.

Of primary interest to this research is the AFASPO letter which requested that programmer resumes for all conversion programmers be provided in the format specified(Figure 1)[Air Force Automated Systems Project Office, 1983]. Each program converted was to be detailed in a program information form describing the program and the effort required to convert it(Figure 2). This data collection effort began in June 1983 and is not due to be completed until late 1985.

The programmer resume form is very straightforward; however, the information requested, in some cases, produces a variety of responses. For this reason, an encoding scheme (presented in Appendix B) was developed by this researcher to refine the raw data.

The program information form is also straightforward but it requires some definition of terms. The "System Code", "DSD"(Data System Designator), "System Title" and "Program Title" are all center specific and pre-defined. The "Difficulty" description categories primarily reflect characteristics of source programs that Sperry-Univac had found to require manual recoding after the automatic recoding of the automatic translator[Air Force Automated Systems Project Office, Undated]. The sum of the number of difficulty categories checked was used as a measure of program difficulty or complexity. The "Switches" category recognizes that B3500 COBOL-68 software switches used in a VALUE statement are not available in COBOL-74. The "Interrogate" category reflects the requirement to use a different method to test for the presence of a disk or diskpack file since the INTERROGATE statement is not available in UNIVAC

I. Programmer Code _____

II. Education

College Graduate? _____ Degree: Associate _____ Bachelors _____
 Masters _____ PhD _____

Major(s): _____ Minor(s): _____

Formal Instruction (in relation to computer field):

III. Background

A. How many years experience do you have in the computer field? _____

B. How many of these years are actual programming years? _____

C. How long have you been programming in COBOL 68? _____
 _____, COBOL 74? _____

D. Was this experience primarily in development or maintenance work? _____

E. If the majority of programs you shall be transitioning are not COBOL, then what type of system are they _____ and how many years experience do you have with them? _____

F. How much experience do you have with transitioning? Explain. _____

G. How much experience do you have in working with Job Control Languages? _____

H. List any other experience which will aid you in transitioning unique software to Phase IV. _____

Figure 1. AFASPO Phase IV Programmer Resume Form

I. General

System Code _____

JSD _____

System Title _____

Program Title _____

Date Started _____

Lines of Code (Start) _____

Date Completed _____

Lines of Code (Finish) _____

II. Program

A. Type (Check One)

☐ Fortran☐ Batch☐ AFOLDS☐ On-line☐ Assembler☐ COBOL☐ Other: _____

B. Difficulty (Check each one the program contains)

☐ Sort☐ Reel I/O☐ Zip☐ Random I/O☐ Switches☐ Copy
(Libraries)☐ Comp Data☐ Interrogate☐ Call

III. Activity

| | Programmer(s) | | | | | |
|-------------------------|---------------|--|--|--|--|--|
| | | | | | | |
| Documentation | | | | | | |
| Data File Transfer | | | | | | |
| ADS Translation | | | | | | |
| Create Control Language | | | | | | |
| Test/Debug | | | | | | |
| Miscellaneous | | | | | | |
| Knowledge Code* | | | | | | |

Manhours
Expended

*Knowledge code is in relation to knowledge of this program. None at all, Severely knowledgeable, Skilled the program.

Figure 2. AFASPO Phase IV Program Information Form

COBOL-74. The "Reel #'s" category refers to the B3500 COBOL-68 USE verb which specifies procedures for tape label handling such as reel number extraction. This capability is not available in COBOL-74; the CALL routine replaces this function. The "Zip" category indicates that the ZIP verb which causes the Burroughs operating system to execute a control instruction contained within the operating object program is not available in COBOL-74 so it must be recoded. The "Sort" category indicates great care is required when sorting is called for because B3500 systems use the EBCDIC character set where alphabetics appear before numerics while the Sperry-Univac systems use the ASCII character set whose collating sequence is the reverse. Logical compares, range checks etc., based on the EBCDIC sequence, have to be recoded. The "Comp Data" category indicates that Comp definitions are manually redefined as numeric fields. This has no affect on arithmetic operations but will cause expansion of record size and data item picture fields. On the B3500 there are two COMP fields per word while the Univac 1100/60 has six fields per word. Consequently, word boundary alignment problems could result if COMP is not redefined. The "Random I/O" category indicates that when random (or sequential)files are open for input/output, the WRITE statement is replaced by a REWRITE statement. Also the entire I/O-Control section is always flagged for manual recoding. The "COPY" category flags the program as requiring access of the COPY library. Subsequent action insures that the COPY executes correctly and the library is available and correctly converted. The "Call" category

indicates the need for routines to be available and converted for the program to execute correctly. Also, the ENTER symbolic statement in Burroughs COBOL-68 which provides the use of an alternate language must be recoded. The COBOL-74 Interprogram Communication module, which provides for CALL and ENTER statements to communicate with other programs, must be used.

The first row of blocks of the activity matrix of the program information form requests the programmer code as specified on the programmer resume. The "Data File Transfer" accounts for the time required to dump program external tables and test data to tape from the old system and its subsequent uploading on the Univac system. This is typically insignificant since several data files may be transferred at once quite easily. The "ADS Translation" accounts for both the automatic and manual recoding of the program ("ADS"). The automatic recoding effort may be quite insignificant since several programs may be grouped and funneled through the automatic translator which may require only a few minutes per program. The "Create Control Language" category accounts for the time, in many cases minimal, to establish the Univac job control statements pertinent to the program.

The program information form contains various program characteristics which the programmer identification code(s) links to programmer attributes. These groups of data provide variables such as those suggested by many of the researchers mentioned in Chapter 3.

The programmer resume and programmer information forms provided the basic data required for the productivity analysis and the corresponding encoded data files are included in Appendix C and D respectively. However, because of the distinction between system level and program level manhour accounting and the nature of cost/effort estimation models, additional data was necessitated. The Air Force Data Systems Design Center(AFDSDC) and the Tactical Air Command(TAC) provided system level manhour data which accounted for most of the second stage(conversion) of the Phase IV effort within each organization. Each record of this additional data, detailed in chapter 6, contains a code for the organization and a system code followed by lines of code, manhours and number of programs.

CHAPTER V

CONVERSION PROGRAMMER PRODUCTIVITY ANALYSIS

Introduction

The Air Force Phase IV software conversion effort data provided the basis for this research. Before initiating the productivity analysis, the programmer resume and the program information data were viewed and studied separately. The separate data files were then integrated, as appropriate, for further detailed analysis. The objective was to study the impact of program factors and programmer attributes on the conversion productivity of programmers. A brief discussion of the preliminary analysis(detailed in Appendix B) seems appropriate before proceeding with the details of the productivity study. A glossary of significant statistical terms used in this thesis is provided as a reference.

Preliminary Analysis

The preparatory stage of the study required the formulation of an encoding scheme for the raw data and the construction of both a programmer resume file and a program information file. These two files were analyzed separately by means of tabular summaries, Chi Square tests and Factor Analysis. The initial Chi Square tests revealed that recoding of some categorical variables was necessary for the tests to be valid so changes to the original encoding scheme were implemented. It was also discovered that most of the program information data showed no man-hours for the first two activities,

documentation and data file transfer, since these two activities were typically handled at the system level. To improve the uniformity of the data, the total effort hours for productivity analysis were derived from summing only four of the six categories of the program information form activity matrix. The details of this preparatory stage and preliminary analysis are contained in Appendix B.

The programmer resume file and program information file were merged to generate a productivity file of records containing both program and associated programmer data. The program conversion effort data, consisting of 130 programs all written in COBOL, came from six different Air Force centers. In order to insure a precise set of data for research where distinct relationships between a programmer's attributes and the conversion effort could be studied, a subset of the productivity sample file was created. This subset only contains programs converted by one programmer since group interaction and group productivity are not considered in this study. The subset only contains 51 programs and is titled the individual productivity file. All work discussed uses the individual productivity file.

Definitions and Assumptions

Two items that have typically caused confusion in research on new software development are lines of code and total man-hours used. For this conversion research, STLOC(starting LOC) is used to measure the size of a program before undergoing the conversion process. Comment lines are included in the count but not job control statements since

no such statements are used by programmers in the B3500 base level environment. SUMHR is used to measure the total number of hours spent by a programmer in recoding a program (including time involved in using the automatic translator), generating job control statements, testing the results, and related miscellaneous hours. As mentioned above, the documentation and data file transfer activities were not included in SUMHR because of inconsistencies in reporting. The SUMHR definition generally parallels that of programming time used by Jeffery and Lawrence[1979]. However, there is no clear definition of what miscellaneous hours a programmer is legitimately authorized to count. The definition of all other variables was included in Appendix B but a brief statement of their meaning is made in the next section of this chapter.

General Overview of Analysis Methodology

The framework for this analysis is a multi-step process embodied in two basic stages. First, the variables, which resulted from the data collection, underwent resolute cerebration which involved an initial separate investigation of the categorical and continuous variables. Secondly, the variables found to be significant were used as the independent variables in regression analysis to develop a model which might be useful in explaining productivity. The productivity measure used as the dependent variable was lines of code per hour(LOCPERHR) calculated by dividing STLOC by SUMHR for each program. Following Jones' suggestion, hours per hundred lines of code(HRPERHLO) was used

as a dependent variable to provide parallel results for comparison [Jones, 1978]. A natural log transformation of LOCPERHR was also regressed as a dependent variable.

Selection of Relevant Variables

To clarify the starting basis for this section of the work, it is appropriate to present a synopsis of the research variables. There are basically two sets of variables: those describing the programmer and others related to the program itself. An additional grouping of the variables is by type; i.e., categorical (nominal) or continuous. These two groupings provided a point of departure for the analysis. A description of all the variables including those subjected to recoding are included in Appendix B.

Categorical Variables Subjected to Analysis of Variance

Two variables of primary interest are DEGREE (level of college education) and MAJOR (academic major). The value of DEGREE was recoded to be one of the following: 0 (no college), 1 (some college) or 2 (college graduate). Those that have attended college and recorded a major fall into one of four categories of MAJOR: 0 (none), 1 (other), 2 (DP-MIS, Math, Science) or 3 (Computer Science). Another variable recoded was one describing the type of conversion experience (CONEXP). CONEXP was permitted to take one of four values: 0 (no experience), 1 (some experience), 2 (greater experience including any COBOL-68 to COBOL-74 experience), and 3 (early Phase IV experience). The values

of programmer experience type(PTYPE) maintained their initial definitions: 0(no experience), 1(maintenance type), 2(maintenance and development type), and 3(mostly development type).

All variables describing an individual's formal training were defined as binary. A value of 1 meant that the individual had training in the specific area of variable definition while a 0 signified no training. The variables in this group are: FTNG6 (COBOL-74), FTNG5(all other COBOL), FTNG4(AF Online Data System), FTNG3(other programmer training), FTNG2(Sperry-Univac training), and FTNG1(other general ADP training).

In the process of learning more about these variables, the average value of LOCPERHR for each categorical value was calculated(Table 3). Then an analysis of variance model was formed with all the above being explanatory variables and LOCPERHR as the dependent or response variable. The SAS general linear model(GLM) procedure was used for the analysis of this exploratory design. A definitional aid for interpreting GLM output is provided in the glossary.

The SAS results demonstrated an analysis of variance(ANOVA) test statistic(model F value) of 1.9 with a significance level of 0.0625 for the entire model. Given the null hypothesis that all the coefficients(or contributions) of the variables equal zero(0), the probability of rejecting a true null hypothesis(Type I error) is 0.063. Since the probability of Type I error is greater than the generally accepted criteria of 0.01 or 0.05, one can not reject the

Table 3. Lines of Code Per Hour(LOC PER HR) Averages for Categorical Variables.

| Variable | N | Mean | Variable | N | Mean | Variable | N | Mean |
|----------|----|-------|----------|----|-------|----------|----|-------|
| MAJOR | | | PTYPE | | | CONEXP | | |
| 0 | 40 | 79.5 | 0 | 10 | 86.3 | 0 | 45 | 63.1 |
| 1 | 9 | 37.1 | 1 | 4 | 20.5 | 1 | 3 | 228.3 |
| 2 | 1 | 24.4 | 2 | 19 | 37.6 | 2 | 0 | - |
| 3 | 1 | 21.4 | 3 | 18 | 105.5 | 3 | 3 | 10.7 |
| DEGREE | | | FTNG1 | | | FTNG2 | | |
| 0 | 23 | 89.5 | 0 | 35 | 86.3 | 0 | 5 | 27.4 |
| 1 | 18 | 63.5 | 1 | 16 | 33.5 | 1 | 46 | 74.4 |
| 2 | 10 | 35.5 | | | | | | |
| FTNG3 | | | FTNG4 | | | FTNG5 | | |
| 0 | 4 | 172.5 | 0 | 50 | 70.7 | 0 | 35 | 86.4 |
| 1 | 47 | 61.0 | 1 | 1 | 21.4 | 1 | 16 | 33.3 |
| FTNG6 | | | | | | | | |
| 0 | 48 | 59.9 | | | | | | |
| 1 | 3 | 228.3 | | | | | | |

null hypothesis which means that either the null hypothesis is true or the data, for some reason, does not allow the detection of small differences from zero. More pointedly, the partial F values for each of the categorical variables were all less than 1 and no level of significance was less than 0.5 meaning that the explanatory contribution of each and every categorical variable was statistically nonexistent; that is, one can not reject the hypothesis that the contribution of each coefficient of the variables is 0.

One reason for these results is the unequal cell frequencies of categorical variables which implies the presence of multicollinearity[Iverson and Norpoth, 1976]. GLM optionally provides for the computations of tolerance values for each variable. A tolerance value is the inverse of the variance inflation

factor(VIF) which measures the combined effect of the dependencies among the regressors on the variance of the term whose VIF is in question[Montgomery and Peck, 1982]. One or more large VIFs indicates multicollinearity and practical experience has demonstrated that if any of the VIFs exceeds 5 or 10, poor estimation of associated coefficients results. Several VIFs in the ANOVA model were much larger than 10 indicating multicollinearity. In pure analysis of variance work, contingency table tests of independence, such as those described in Appendix B, are typically used to detect multicollinearity[Iversen and Norpoth, 1976]. Only when the Chi-Square value equals 0 is one assured of uncorrelated variables.

Of the methods for dealing with multicollinearity, variable elimination seemed to be the most practical[Montgomery and Peck, 1982]. This opened up the entire subject of variable selection techniques and procedures where one can find clear signs of Mendenhall's exposition that "the application of theory to the solution of practical problems is an art and subject to debate[Mendenhall, 1968]." Cox and Snell provide some basic guidelines, useful for choosing variables in observational studies, that are applicable to this work[Cox and Snell, 1974].

One criterion for choosing variables is for the available data to be roughly evenly split across the levels of a categorical variable in order for the effect on the response variable to be as clear-cut as possible. Variables which indicate high dependencies and/or are considered alternatives are good candidates for

elimination because of multicollinearity and for reasons of simplicity. Including too many variables in a model typically raises the mean square error of prediction. If the objective of a fit to data is primarily for prediction then two different models, involving different variables, are equally acceptable if they fit the data equally well. Simplicity, in terms of the number of variables, is the basis for choosing between these models, if necessary.

In terms of an adequate split of the data over the levels of a variable, the best candidates for selection were DEGREE(level of college education), PTYPE(programmer type), FTNG1(general ADP training), and FTNG5(general COBOL training). A possible second place selection was MAJOR(academic major), with CONEXP(conversion experience type) barely squeezing in for consideration because of its natural significance in this study. Since the Chi-Square tests recorded in Appendix B showed a significant level of association between CONEXP and PTYPE, CONEXP and DEGREE, and CONEXP is really inadequate in its data splitting over its levels, CONEXP was subsequently excluded from consideration. Because both DEGREE and PTYPE exhibited a significant association with FTNG1 and it measures a formal training area of little concern to this study, FTNG1 was also excluded. Since FTNG5 is significantly associated with PTYPE and duplicates a couple of the continuous variables (C68EXP and PGMEXP) and since COBOL training may already be associated with the programmer's academic education, FTNG5 was deleted as a candidate variable. Even though MAJOR and DEGREE showed a highly significant

association, MAJOR was allowed to remain as a candidate variable to provide for correspondence with other research. Care was to be exercised in interpreting model results if both remained in the final reduced model. In summary, the categorical variables chosen as candidates for the combined model were PTYPE, DEGREE and MAJOR.

Continuous Variables Scrutinized

The continuous variables derived from the programmer resume file include total years of ADP experience(TOTEXP), years of programmer experience(PGMEXP), COBOL-68 years of experience(C68EXP), COBOL-74 years of experience(C74EXP), and years of JCL experience(JCLEXP). The productivity file also includes continuous variables from the program information file. Of the two measures of lines of code, STLOC(starting LOC) is the most appropriate for a predictive model since FILOC(finishing LOC) is not initially available. The level of difficulty or complexity of a program is significant for productivity studies so a count of the number of difficulty categories checked by the programmer(SUMDIF) was used in addition to lines of code(STLOC) to further define the program. The knowledge code(KCA) of the programmer for a specific program was reserved as a continuous variable since it is an ordered variable which could very well have intermediate values.

The process of selecting candidate independent continuous variables follows the same basic pattern as that for categorical variables. Multicollinearity, duplication of a variable, inadequacy

of data, and simplicity were used as criteria for candidate variable selection. From the factor analysis and correlation matrix of the preliminary analysis(Appendix B), one can vividly see the presence of correlation between the experience variables. TOTEXP, PGMEXP and C68EXP seem to provide very similar information and thus may not all be required in a model. C74EXP and JCLEXP show some correlation but not of the same degree as the other three variables.

An advantage of continuous variables is the availability of automated techniques to assist one in variable selection. Stepwise regression techniques have slight differences in their approach and suitability. Montgomery and Peck stated that the backward elimination algorithm is often less adversely affected by the correlative structure of the regressors; therefore, it was chosen for this variable selection[Montgomery and Peck, 1982]. The backward elimination algorithm permits a variable to return to the mix (forward selection) if it becomes significant after other variables leave the mix(backward elimination). As suggested by Montgomery and Peck, the partial F test level of significance for forward selection was set at 0.25 while the level of significance for backward elimination was set at 0.10[Montgomery and Peck, 1982]. The dependent variable was again LOCPERHR.

Using the SAS STEPWISE procedure for Backward Elimination, only PGMEXP and TOTEXP were deleted from the list of variables with all the remaining ones(STLOC, KCA, SUMDIF, JCLEXP, C68EXP and C74EXP) being significant at the 0.007 level or less. Montgomery and Peck

warn that, though helpful, stepwise techniques like backward elimination do not necessarily produce an optimal model since there may be several models that are equally good. With this in mind and to expand the scope of explanatory variables for the model, PGMEXP was also included as a regressor to bring in the individual's general and overall programming experience.

Model Variables Selected

In summary, the categorical variables of interest for starting the regression analysis were DEGREE(level of college education), PTYPE(programmer type), and MAJOR(academic major). The continuous variables for the initial model were STLOC(starting LOC), KCA(programmer's knowledge level), SUMDIF(measure of program difficulty), C68EXP(years of COBOL-68 experience), PGMEXP(years of general programming experience), C74EXP(years of COBOL-74 experience) and JCLEXP(years of experience with JCL).

Model Specification and Analysis

When building a model with both quantitative and qualitative variables, one is in the statistical realm of analysis of covariance which is a special case of the linear model. However, when quantitative(covariate) and categorical independent variables are of equal interest or the design is unbalanced, regression analysis is not only equivalent to analysis of covariance it is also more appropriate[Ahtola and Wildt, 1978]. Equivalency is obtained by

representing the qualitative variables by dummy variables. This is easily done within the SAS GLM procedure by identifying the categorical variables in the CLASS statement. The glossary contains an aid for interpreting GLM output as presented in table form in this chapter.

Initial Model Analysis

When performing the initial runs, it was discovered that GLM normally sets the coefficient of the highest level of a categorical variable, whether alphabetical or numerical categories, to zero and includes its effect or value in the intercept term. Therefore, for the purpose of the regression analysis the codes for PTYPE, DEGREE and MAJOR were reversed with the highest value becoming 1 and so on, to allow these lower numbered levels, which were of greater significance, to have specifically defined coefficients.

The initial model, with LOCPERHR as the dependent variable and STLOC, KCA, SUMDIF, C68EXP, PGMEXP, JCLEXP, C74EXP, DEGREE, MAJOR and PTYPE as the independent variables is shown in Table 4. The coefficient of determination(R^2) shows that 77% of the variability of LOCPERHR is explained by this model. The coefficient of variation(C.V.) expresses the unexplained deviation remaining in the data relative to the mean response. The mean square error(MSE) is used as an estimate of the model's variance yielding root MSE as an estimate of the model's standard deviation. The model's F value is significant at 0.0001 indicating that at least one coefficient is

Table 4. Initial Version of Productivity(LOCPERHR) Model.

| DEPENDENT VARIABLE: LOCPERHR | | | | |
|------------------------------|--------|----------------|-------------|---------------|
| SOURCE | DF | SUM OF SQUARES | MEAN SQUARE | |
| MODEL | 14 | 232050 | 16575 | |
| ERROR | 36 | 70551 | 1959 | |
| CORRECTED TOTAL | 50 | 302602 | | |
| MODEL F | PR > F | R-SQUARE | C.V. | LOCPERHR MEAN |
| 8.46 | 0.0001 | 0.77 | 63.5 | 69.76 |
| SOURCE | DF | TYPE III SS | F VALUE | PR > F |
| STLOC | 1 | 107658 | 54.93 | 0.0001 |
| KCA | 1 | 9048 | 4.62 | 0.0385 |
| C68EXP | 1 | 16899 | 8.62 | 0.0058 |
| SUMDIF | 1 | 28862 | 14.73 | 0.0005 |
| PGMEXP | 1 | 11120 | 5.67 | 0.0226 |
| JCLEXP | 1 | 29339 | 14.97 | 0.0004 |
| C74EXP | 1 | 1115 | 0.57 | 0.4557 |
| DEGREE | 1 | 195 | 0.10 | 0.7541 |
| MAJOR | 2 | 3272 | 0.83 | 0.4422 |
| PTYPE | 3 | 10426 | 1.77 | 0.1697 |

non-zero; that is, the probability of a Type I error of rejecting the null hypothesis of zero coefficients is 0.0001. The partial Fs, or partial sums of squares, indicate that four variables have a level of significance much higher than the 0.10 guide used for hypothesis testing in regression analysis. Of the four variables, C74EXP, DEGREE, MAJOR and PTYPE, the two with the highest probability or level of significance(DEGREE = 0.75 and C74EXP = 0.46) were eliminated from the model. The VIFs, tolerance values, indicated no significant level of multicollinearity.

Reduction of Initial Model

In deleting insignificant variables from a model, a decrease in R^2 is certain to occur. The R^2 for the subset regression model must be subjected to an adequacy test to insure a "satisfactory" value. Aitkin proposed the calculation of an R^2 -adequate(R_O^2) value as follows[Aitkin, 1974]:

$$R_O^2 = 1 - (1 - R_x^2)(1 + d_{n,p}^a)$$

where

$$d_{n,p}^a = \frac{P(F_{p,n-p-1}^a)}{n - p - 1}$$

R_x^2 = R^2 of initial starting set of variables
 p_x = number of parameters
 n = number of observations
 α = level of significance

The level of significance should be small(say .01-.10) if the exclusion of all irrelevant variables is important. Aitkin also suggested that a larger α (say .25-.50), insured that "important" variables were not excluded while the inclusion of possibly irrelevant variables was tolerated. For this work, an α of 0.25 seemed appropriate since many irrelevant variables had already been excluded. Therefore, with $n=51$, $p=10$ and $\alpha=.25$,

$$d_{51,10}^{0.25} = \frac{10(F_{10,40}^{0.25})}{40} = .4(1.33) = 0.532$$

and

$$R_O^2 = 1 - (1 - 0.77)(1 + 0.532) = 0.65$$

With $R^2 = 0.65$ as a guide, one can begin to review reduced or subset regression models. The next iteration of the model removed C74EXP and DEGREE. The MSE decreased slightly and R^2 was reduced by

only .01 while the model F value increased to 10.01 and the C.V. changed only by about 0.7. All of the variables were significant at .003 or less with the exception of MAJOR which showed a low partial F value and a significance level of 0.307; that is, one can not reject the hypothesis that the coefficient of MAJOR is zero.

Deleting MAJOR from the set of variables, produced a final LOCPERHR model (Table 5) with all variables significant at 0.0006 or less. This model has an MSE lower than that of the initial model and an R^2 of 0.74 which is more than adequate by Aitkin's criteria and only 3% less than the initial model. Table 5 also contains the coefficients associated with all the variables. A discussion of the resulting equation is necessary not only for explanatory purposes but also to verify the model's adequacy by affirming the reasonableness of the results.

The coefficient for STLOC seems small in comparison to the others but realizing that the average STLOC value is over 1000 makes it quite reasonable. The positive impact of size on productivity agrees with the findings of Paulsen, Basili and Freburger, and Jeffery and Lawrence in the new development arena [Paulsen, 1981; Basili and Freburger, 1981; Jeffery and Lawrence, 1979]. It is typically counter-intuitive for productivity to increase with lines of code; however, the sample data indicates that this result occurs when most of the code is converted automatically. Other conversion efforts may not support this result especially if only a small percentage of the code is converted automatically.

Table 5. Final Version of LOCPERHR Model.

| DEPENDENT VARIABLE: LOCPERHR | | | | |
|------------------------------|----------|----------------|-------------|---------------|
| SOURCE | DF | SUM OF SQUARES | MEAN SQUARE | |
| MODEL | 9 | 222735 | 24748 | |
| ERROR | 41 | 79867 | 1948 | |
| CORRECTED TOTAL | 50 | 302602 | | |
| MODEL F | PR > F | R-SQUARE | C.V. | LOCPERHR MEAN |
| 12.7 | 0.0001 | 0.74 | 63.3 | 69.76 |
| SOURCE | DF | TYPE III SS | F VALUE | PR > F |
| STLOC | 1 | 120067 | 61.64 | 0.0001 |
| KCA | 1 | 35961 | 18.46 | 0.0001 |
| C68EXP | 1 | 43080 | 22.12 | 0.0001 |
| SUMDIF | 1 | 28152 | 14.45 | 0.0005 |
| PGMEXP | 1 | 27230 | 13.98 | 0.0006 |
| JCLEXP | 1 | 54250 | 27.85 | 0.0001 |
| PTYPE | 3 | 43238 | 7.40 | 0.0005 |
| PARAMETER | ESTIMATE | | | |
| INTERCEPT | 64.538 | | | |
| STLOC | 0.132 | | | |
| KCA | -20.630 | | | |
| C68EXP | 20.528 | | | |
| SUMDIF | -27.750 | | | |
| PGMEXP | 21.703 | | | |
| JCLEXP | -46.786 | | | |
| PTYPE 1 | 16.223 | | | |
| PTYPE 2 | -68.289 | | | |
| PTYPE 3 | -82.803 | | | |

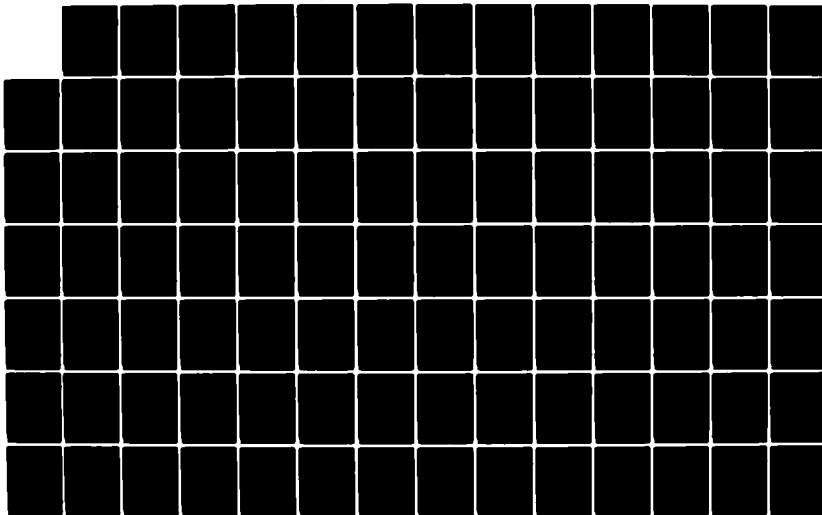
AD-A145 757

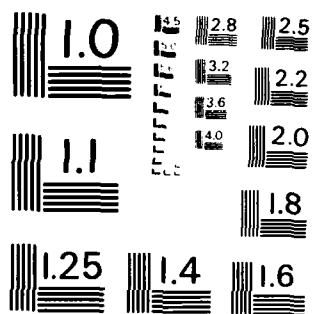
A METHODOLOGY FOR THE ANALYSIS OF PROGRAMMER
PRODUCTIVITY AND EFFORT ESTI... (U) AIR FORCE INST OF
TECH WRIGHT-PATTERSON AFB OH J D FERNANDEZ MAY 84
AFIT/CI/NR-84-440 F/G 9/2

2/3

UNCLASSIFIED

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

The coefficient of KCA(programmer's knowledge of the program) initially seemed to be an error. Intuitively, if a programmer is more familiar with a program(higher KCA), it should be expected that his productivity would be higher. However, this opposite result confirms the comments made by Oliver about a U.S. Navy conversion which indicate that if a programmer is converting his own program, he is more likely to modify and/or correct program lines rather than just recode the original program[Oliver, 1978]. Programs that fall in this category sometimes show that the finishing lines of code are the same or less than the starting lines of code, which is extremely unlikely in this conversion as discussed in chapter 4. Programmers may be "improving" their own programs or implementing unauthorized changes requested by local management.

The coefficient of the variable SUMDIF, measure of program difficulty, results in a negative value. It is intuitively logical for this to be true since one would expect the productivity to decrease as the level of difficulty of the program increases.

The presence of both C68EXP(COBOL-68 experience) and PGMEXP(general programming experience) initially led to a concern for multicollinearity. However, the VIFs(variance inflation factors) indicated no significant level of multicollinearity. An increase of 20 LOCPERHR for each year of C68EXP and each year of general programmer experience(PGMEXP) is possible for the conversion realm. The magnitude of these coefficients is applicable to this case study only; however, this result generally agrees with several new

development studies which showed that experience significantly influences development time.

The negative coefficient for JCLEXP is only reasonable because it was discovered during the data collection that the experience in many cases included primarily academic experience for junior programmers or programmer trainees. In addition, JCL experience for intermediate or senior programmers in the B3500 environment was typically nil since there is no JCL for associated application programs. JCLEXP is essentially an indicator for an entry level programmer and nothing more.

The categorical variable of PTYPE(programmer type) has four levels but only three coefficients since the "no experience" level effect is included in the intercept term by the GLM procedure. It is very reasonable for the development type(1) programmer to have a +16 coefficient since he is more likely to have the highest productivity because he probably has the greatest expertise. Programmers with both development and maintenance experience(type 2) exhibit a negative effect on LOCPERHR. This seems reasonable because programmers in this category do not have the depth of experience as development programmers since this category seems to be a stepping stone for entry-level programmers who upgrade from the maintenance type(3) level. The data shows, what may be common in many organizations, that the inexperienced programmers typically start out in maintenance.

Model adequacy is measured by several means, including R^2 and the reasonableness of the resulting parameters. Draper and Smith and others suggest that another measure of model adequacy is that the observed or model F value be about four times the critical F value [Draper and Smith, 1966]. Since the fitted model exhibited an F value of 12.7 and the critical F value for an alpha of .05 is 2.12, the observed or model F value is approximately six times the critical F value. This test suggests strongly that the model is statistically adequate. A final measure of adequacy is the plot of the calculated LOCPERHR(yhat) versus the residuals (difference between observed and predicted value). The residual plot indicated an adequate scatter of the points around 0, thus contributing to the assessment of an adequate model.

Alternate Dependent Variable Models

The dependent variable of hours per hundred lines of code converted (HRPERHLO) was used in regression analysis with the same initial set of independent variables used with LOCPERHR. HRPERHLO is essentially a productivity measure of "cost units" as suggested by Jones [1978]. Therefore, the more productive a programmer is the lower the value of HRPERHLO and vice versa. STLOC was converted to HSTLOC (hundreds of lines of code) since this measure is more appropriate for an HRPERHLO model.

The final or subset regression model with all variables significant at the 0.048 level or below is shown in Table 6. The model R^2 of 0.73 is slightly less than that of the LOCPERHR model.

Table 6. Final HRPERHLO Alternate Productivity Model.

| DEPENDENT VARIABLE: HRPERHLO | | | | |
|------------------------------|----------|----------------|-------------|---------------|
| SOURCE | DF | SUM OF SQUARES | MEAN SQUARE | |
| MODEL | 11 | 648.80 | 58.98 | |
| ERROR | 39 | 245.80 | 6.30 | |
| CORRECTED TOTAL | 50 | 894.58 | | |
| MODEL F | PR > F | R-SQUARE | C.V. | HRPERHLO MEAN |
| 9.36 | 0.0001 | 0.73 | 66.8 | 3.76 |
| SOURCE | DF | TYPE III SS | F VALUE | PR > F |
| KCA | 1 | 392.78 | 62.32 | 0.0001 |
| HSTLOC | 1 | 93.51 | 14.84 | 0.0004 |
| C68EXP | 1 | 126.37 | 20.05 | 0.0001 |
| SUMDIF | 1 | 26.23 | 4.16 | 0.0482 |
| JCLEXP | 1 | 52.90 | 8.39 | 0.0061 |
| MAJOR | 3 | 147.53 | 7.80 | 0.0003 |
| PTYPE | 3 | 143.26 | 7.58 | 0.0004 |
| PARAMETER | ESTIMATE | | | |
| INTERCEPT | 1.1264 | | | |
| KCA | 3.2682 | | | |
| HSTLOC | -0.3890 | | | |
| C68EXP | -2.2135 | | | |
| SUMDIF | 1.0864 | | | |
| JCLEXP | 1.6393 | | | |
| MAJOR 1 | -13.2499 | | | |
| MAJOR 2 | -15.6522 | | | |
| MAJOR 3 | 7.8853 | | | |
| PTYPE 1 | -0.0134 | | | |
| PTYPE 2 | -7.2428 | | | |
| PTYPE 3 | 5.5494 | | | |

The observed model F value of 9.36 is 4.6 times the critical F value of 2.04(F.05, 11, 39) which qualifies as an adequate measure yet is lower than that of the LOCPERHR model. As expected because the dependent variable is in cost units, the coefficients for this inverse productivity(cost) model have signs opposite of those in the LOCPERHR model. This is true for KCA, HSTLOC, C68EXP, SUMDIF and JCLEXP. Notice, however, that PGMEXP was found to be insignificant and thus left the model while MAJOR remained at a fairly significant level. The coefficient associated with a CS major(type 1) is negative as expected since a reduction in HRPERHLO seems reasonable for such a major. The DP-MIS/Math/Engineering category(type 2) also shows a negative effect on HRPERHLO. The relatively small difference in the absolute value might be explained by noting that the CS academic major is somewhat new and more likely for newer and less experienced programmers. Intuitively, it seems reasonable to expect a social science, business or other major(type 3) to exhibit a positive effect on HRPERHLO.

To reduce some of the variation in the data, an alternate dependent variable model, with the natural log of LOCPERHR or LOGLOCPH, was studied as suggested by Montgomery and Peck[1982]. Table 7 presents the final subset model that resulted from the regression analysis. The variables which remained in the model exhibited a level of significance of the variables of 0.008 or lower. The coefficient of variation does show a decrease while the R^2 only increases by .01. The observed model F value of 10.89 is 5.3 times

Table 7. Final LOGLOCPH Alternate Productivity Model.

| DEPENDENT VARIABLE: LOGLOCPH | | | | |
|------------------------------|----------|----------------|-------------|---------------|
| SOURCE | DF | SUM OF SQUARES | MEAN SQUARE | |
| MODEL | 11 | 38.1400 | 3.4673 | |
| ERROR | 39 | 12.4228 | 0.3185 | |
| CORRECTED TOTAL | 50 | 50.5628 | | |
| MODEL F | PR > F | R-SQUARE | C.V. | LOGLOCPH MEAN |
| 10.89 | 0.0001 | 0.75 | 14.9 | 3.7715 |
| SOURCE | DF | TYPE III SS | F VALUE | PR > F |
| KCA | 1 | 13.224 | 41.52 | 0.0001 |
| STLOC | 1 | 13.882 | 43.58 | 0.0001 |
| C68EXP | 1 | 5.607 | 17.60 | 0.0002 |
| SUMDIF | 1 | 2.438 | 7.66 | 0.0086 |
| JCLEXP | 1 | 3.505 | 11.00 | 0.0020 |
| MAJOR | 3 | 4.334 | 4.54 | 0.0080 |
| PTYPE | 3 | 5.911 | 6.19 | 0.0015 |
| PARAMETER | ESTIMATE | | | |
| INTERCEPT | 4.15786 | | | |
| KCA | -0.59968 | | | |
| STLOC | 0.00150 | | | |
| C68EXP | 0.46624 | | | |
| SUMDIF | -0.33126 | | | |
| JCLEXP | -0.42195 | | | |
| MAJOR 1 | 1.75704 | | | |
| MAJOR 2 | 2.79148 | | | |
| MAJOR 3 | -1.32880 | | | |
| PTYPE 1 | 0.20660 | | | |
| PTYPE 2 | 0.86744 | | | |
| PTYPE 3 | -1.67892 | | | |

the critical F (2.04) which is adequate but less than that of the LOCPERHR model. The resulting set of variables is the same as that of the HRPERHLO model and only one variable different from the LOCPERHR model, with MAJOR replacing the variable PGMEXP. As would be expected, the signs of the coefficients are exactly the opposite of those of the HRPERHLO model and parallel those of the LOCPERHR model. This model might be considered equally as good as the LOCPERHR except for the fact that the model uses logs rather than natural values thus complicating the interpretation of the results. The coefficient of variation(C.V.) does show a significant reduction in the unexplained deviation remaining in the data since logs were used in this model.

Consideration of Organizational Impact

Paralleling the new development productivity work of Jeffery and Lawrence, it was decided to include the organization or conversion center(MAJCOM) in the final LOCPERHR model to study the results[Jeffery and Lawrence, 1979]. Initially, the R^2 increased to 0.79; however, the PTYPE variable was found to be insignificant at a 0.3 level and MAJCOM only marginally significant at 0.09. PTYPE was removed from the model producing a further reduced subset with C68EXP insignificant at a level of 0.1415. Table 8 shows the final LOCPERHR with MAJCOM added and PTYPE and C68EXP removed.

The R^2 of .75 for the organizational LOCPERHR model is only .01 higher than the original LOCPERHR model without MAJCOM. There is

Table 8. Final LOCPERHR Model With Organization.

| DEPENDENT VARIABLE: LOCPERHR | | | | |
|------------------------------|----------|----------------|-------------|---------------|
| SOURCE | DF | SUM OF SQUARES | MEAN SQUARE | |
| MODEL | 9 | 227237 | 25249 | |
| ERROR | 41 | 75365 | 1838 | |
| CORRECTED TOTAL | 50 | 302602 | | |
| MODEL F | PR > F | R-SQUARE | C.V. | LOCPERHR MEAN |
| 13.74 | 0.0001 | 0.75 | 61.5 | 69.76 |
| SOURCE | DF | TYPE III SS | F VALUE | PR > F |
| STLOC | 1 | 108053 | 58.78 | 0.0001 |
| KCA | 1 | 9341 | 5.08 | 0.0296 |
| SUMDIF | 1 | 45253 | 24.62 | 0.0001 |
| PGMEXP | 1 | 26888 | 14.63 | 0.0004 |
| JCLEXP | 1 | 47236 | 25.70 | 0.0001 |
| MAJCOM | 4 | 82698 | 11.25 | 0.0001 |
| PARAMETER | ESTIMATE | | | |
| INTERCEPT | 64.300 | | | |
| STLOC | 0.138 | | | |
| KCA | -20.187 | | | |
| SUMDIF | -33.378 | | | |
| PGMEXP | 19.730 | | | |
| JCLEXP | -39.567 | | | |
| MAJCOM F | 115.751 | | | |
| MAJCOM J | 16.089 | | | |
| MAJCOM S | -64.195 | | | |
| MAJCOM T | 29.444 | | | |

also a slight increase in the model F value, from 12.7 to 13.7. The coefficients of the variables, common to both models, have the same signs and approximately the same magnitude. Since the intercept is almost exactly the same, it is apparent that the MAJCOM has replaced the combined effect of PTYPE and C68EXP. By reviewing the MAJCOM coefficients, one can see real differences in productivity among the organizations suggesting agreement with researchers who exposit the inclusion of organization in productivity research. The fact that the original LOCPERHR model appears equally adequate with this revised model, leads one to choose the original LOCPERHR model as the preferred model for continuing the analysis.

Model Validation

Montgomery and Peck list two basic validation procedures that could be applied to this work [Montgomery and Peck, 1982]. One, collecting fresh data to investigate the model's predictive performance could be done as future work when more data is available but is not feasible at this time. The second, data splitting, has various ways of being applied. The most common data splitting technique is setting aside some of the original data and using these observations to investigate the model's predictive performance. Since the sample was not very large, this technique was not feasible; however, the prediction error sum of squares or PRESS statistic may be considered as a form of data splitting. To calculate PRESS, an observation i is deleted and a regression model is fitted to the remaining $n-1$ observations using

the resulting equation to predict the withheld observation dependent variable, say y_i . Denoting this predicted value \hat{y}_i , one can calculate the prediction error for point i as $e_i = y_i - \hat{y}_i$. Then the PRESS statistic is defined as the sum of squares of the resulting "deleted residuals":

$$\text{PRESS} = \sum_{i=1}^n e_i^2$$

The GLM procedure calculated the PRESS statistic for the model with a value of 143986. Montgomery and Peck suggest an R^2 for prediction to validate the model results:

$$R^2_{\text{Prediction}} = 1 - \frac{\text{PRESS}}{\text{Corrected Sum of Squares}}$$

For the LOCPERHR model,

$$R^2_{\text{Prediction}} = 1 - \frac{143986}{302602} = 1 - 0.476 = 0.524$$

Therefore, one could expect this model to "explain" about 52.4 percent of the variability in predicting new observations, as compared to the 74 percent of the variability in the original data explained by the least squares fit. The difference points to the great variability of the raw data indicating that though the model may be adequate, additional sample data would be useful for further regression analysis and model enhancement.

Wolberg's model for productivity in terms of lines per day was used to calculate productivity in an attempt to determine an R^2 for prediction for comparison with the validation results above [Wolberg,

1983]. Assuming 7.88 hours per day (Wolberg uses 173.3 hours/month and 22 days/month), Wolberg's model becomes:

$$P_{LH} = 0.808 * L^{0.53}$$

where P_{LH} is lines of code per hour and L is equal to STLOC. Since Wolberg's model is based on data over the entire spectrum of the conversion project, the model was arbitrarily adjusted with a multiplier based on an assumption that the recoding and unit testing activities involved 50 percent of the total hours. This required an adjustment as follows:

$$P_{LH} = 2 * 0.808 * L^{0.53} = 1.616 * L^{0.53}$$

Montgomery and Peck suggest the following formula for R^2 for prediction with new data:

$$R^2_{\text{Prediction}} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

where \hat{y}_i is the predicted value for y_i and \bar{y} is the mean of the y_i values and the summations are done over the n data points available.

Wolberg's adjusted productivity model produced an R^2 for prediction of 0.37 when it was assumed that programming activities were 50% of the original overall time. Adjusting Wolberg's model with a multiplier of 2.5 (assuming programming activities to be 40% of the original time) produced an R^2 for prediction of 0.51. Finally, a multiplier of 3.33 was used to adjust the model and the R^2 for

prediction increased again to 0.63. These results seem to indicate that with certain arbitrary multipliers being applied to Wolberg's model, one could have a productivity model comparable in adequacy to the LOCPERHR regression model developed above. However, with only one parameter in Wolberg's model, much of the variability in the data may not be captured. This would lead one to continue to give preference to the LOCPERHR model.

CHAPTER VI

SOFTWARE CONVERSION EFFORT ESTIMATION ANALYSIS

Introduction

As evidenced by the literature review presented earlier, little work has been done in the area of effort estimation associated with conversion of software. The purpose of the last phase of the research was to validate the accuracy of significant models or methods for estimating effort and to develop a model for the environment under study. The basis for the validation was again the Air Force Phase IV data.

This particular endeavor required obtaining conversion effort hours at the system, rather than program, level. As discussed in the last chapter, there were a couple of program related conversion activities that were reported at the system level, or not reported at all. Thus, the two leading Phase IV organizations, the Air Force Data Systems Design Center (AFDSDC) and the Tactical Air Command(TAC), were contacted for additional data. These two organizations, being almost 100% complete, were thus able to provide more accurate system level effort data for this analysis. The data included the number of hours, number of programs and LOC for each system completed and is provided in Table 9.

The approach to the study involved the formulation of some basic assumptions needed for the detailed specification of the models to be used. All the models were converted to the uniform basis of hour

Table 9. Phase IV System Level Effort Data.

| MAJCOM System | | LOC | Nr. of Programs | Hours |
|---------------|----|-------|-----------------|-------|
| O | AX | 1985 | 2 | 940 |
| O | BK | 8632 | 6 | 1396 |
| O | CB | 1484 | 1 | 642 |
| O | EB | 7571 | 8 | 1004 |
| O | IE | 6918 | 5 | 1017 |
| O | IN | 37873 | 11 | 2793 |
| O | LY | 7506 | 8 | 1062 |
| O | QP | 19000 | 13 | 1832 |
| O | SF | 4360 | 1 | 1051 |
| O | ZG | 48227 | 39 | 5268 |
| O | ZP | 2042 | 3 | 856 |
| T | CO | 13910 | 7 | 1078 |
| T | EC | 717 | 1 | 149 |
| T | KV | 982 | 2 | 225 |
| T | MQ | 530 | 1 | 217 |
| T | OT | 6325 | 2 | 641 |
| T | OZ | 33635 | 22 | 2375 |
| T | QF | 1463 | 1 | 448 |
| T | WE | 17156 | 7 | 1112 |
| T | WL | 1569 | 3 | 408 |

estimation, rather than man-month or day estimation. Using data known prior to the conversion(STLOC and number of programs), effort hours required by each system were estimated by each model. These hours were compared to the actual hours and the R^2 for prediction, average squared prediction error(comparable to the mean square error), and average residual(average deviation of actual hours minus predicted hours) were calculated to measure accuracy. The objective is to obtain a relatively high value(close to 1.0) of R^2 for prediction, an average squared prediction error that is as low as possible and an average residual that is relatively close to zero.

Since R^2 and the average residual both have specific numeric goals, they were of primary interest in the analysis. Some tuning of the models was performed and the process repeated. Finally, regression analysis was used to develop a model for the Phase IV environment. The glossary contains definitions of statistical terms which may be helpful in interpreting results.

One assumption made in transforming models to hour estimation was to use eight(8) hours per day and 22 days per month if no hour figures were provided by the model developer. Since only the effort involved in recoding, including related testing and documentation etc., was of interest, only the applicable parts of models providing for other activities were used for this research. The measure of the documentation status, required by some of the models, is assumed to be 90% which seemed appropriate for Phase IV. Other assumptions made for specific models are discussed in the applicable sections of this chapter.

Respecification of Effort Estimation Models

The following models were studied for accuracy: FCSC, Hahn and Stone, Grim, Epler and Andrus, Wolberg, and Basili and Freburger[Federal Conversion Support Center, 1982c; Hahn and Stone, 1970; Grim, Epler and Andrus, 1978; Wolberg, 1983; Basili and Freburger, 1981]. The model suggested by the AFASPO is just a slightly modified version of the Hahn and Stone model so it is not included here[Air Force Automated Systems Project Office, 1982a]. The Basili and Freburger

model is the only one not developed for conversion effort estimation. Details of the models are included in Appendix A.

FCSC Cost Model

The FCSC model was developed for the total coverage of a project, from planning to final implementation[Federal Conversion Support Center, 1982c]. Only those portions of the model specifically related to a programmer's conversion efforts were used. The first function specified in the model that is applicable to this work is that of test data generation. As stated in the model, if the percentage of code the test data is required to exercise equals the percentage of code the test data exercised in the original environment, then about one day per program(E_{TD}) is necessary to validate and transfer data.

For the function of application program and system software conversion, a complexity class must be initially chosen. Class 4, simple syntax translation, appeared to be the best choice for Phase IV. This class provides the design effort(DE) parameter of 1, the programming effort(PE) parameter of .5 and the testing effort(TE) parameter of 2. The documentation status(DOC) for Phase IV was assumed to be .9. The FCSC baseline productivity rate(BR) of 12.7 LOC per day for new development and the measure of total effort(NDE) for new development(100) were also used to calculate the manual daily conversion productivity rate(MCPR):

$$MCPR = \frac{BR * NDE}{[(1.0 - (DOC/2)) * DE] + PE + TE}$$

$$MCPR = \frac{12.6*100}{[(1.0 - (.90/2))*1] + .5 + 2}$$

$$MCPR = 413$$

Then, using the FCSC assumed automatic translator daily conversion productivity rate(ACPR) of 630 LOC, the percentage of code typically translated(T) automatically for Phase IV programs(90%), and STLOC to represent the starting lines of code, the effort required for application program and system software conversion(E_{SW}) may be calculated as:

$$E_{SW} = \frac{STLOC(1 - T)}{MCPR} + \frac{STLOC*T}{ACPR}$$

$$E_{SW} = \frac{STLOC(1-0.9)}{413} + \frac{STLOC*.9}{630}$$

$$E_{SW} = 1.67*STLOC*10^{-3}$$

The function of data file and data base conversion requires the selection of a data complexity class. Class D, appropriate for Phase IV files, provides a file conversion complexity factor (FCF) of 1. Since there are no data description or dictionary languages, the effort for file conversion(E_{DF}) may be calculated as follows:

$$E_{DF} = (F*FCF)*(1.0 - (DOC/2))$$

where

$$F = \text{numbers of files}$$

The number of files per program or system was not readily available for inclusion in the FCSC formula. An estimated average number of files per program was derived from the one source of such information available. An AFSDC planning document shows approximately six files

per program so $6 \cdot P$ was substituted for the number of files, with P representing number of programs in the system[Air Force Data Systems Design Center, 1982]. Therefore,

$$E_{DF} = (6 \cdot P \cdot 1) \cdot (1.0 - (.90/2))$$

$$E_{DF} = 3.3 \cdot P$$

There is no operation control language or JCL for the B3500 systems which make up the sample, thus the FCSC suggested approach of using the application program and system software procedure for estimation does not apply. Since JCL formats are somewhat standard, an estimate of between one half to one day per program seemed appropriate for setting up the respective control language; therefore, three fourths of a day per program was chosen for this effort(E_{CL}):

$$E_{CL} = 0.75 \cdot P$$

This appeared to parallel the 5.3 hour(0.66 day) per program average for Phase IV.

For the function of system testing, it was necessary to again use the unique AFSDC data, mentioned above, to determine the average number of runs(J) per program which was calculated to be two runs per program. The number of files(F) per program was previously calculated to be 6. The FCSC assumes about 5 to 10 reruns(RE) are typically required for conversion testing. For a fairly compatible environment, an RE factor of 5 seems appropriate. The testing effort(E_{ST}) required for each system can then be calculated:

$$E_{ST} = [(1 + F)/10 + ((P + F + J)/80)] \cdot [1 + RE/10]$$

$$E_{ST} = [(1 + 6 \cdot P)/10 + ((P + 6 \cdot P + 2 \cdot P)/80)] \cdot [1 + 5/10]$$

$$E_{ST} = 1.069 * P + .15$$

The function of acceptance testing requires the estimation of the duration(DUR) of the test. A DUR of 1 day per program seems more than adequate for this environment. Using the average of 6 files per program(F), calculated above, one can formulate the equation for acceptance testing effort(E_{AT}) per system as follows:

$$E_{AT} = [DUR * 1/20] + [((P + F)/5) * (1 - e^{-(DUR/20)})]$$

$$E_{AT} = [1 * P * 1/20] + [((P + 6 * P)/5) * (1 - e^{-(1 * P/20)})]$$

$$E_{AT} = 1.45 * P - 1.4 * P * e^{-(P/20)}$$

The redocumentation activity formula requires the estimation of the coordination effort (RCOR) between staff members during redocumentation. Since Phase IV systems were well documented, FCSC's typical RCOR factor of 10% seemed adequate. One can calculate the amount of time(E_{RD}) for programmers to redocument a system as follows:

$$E_{RD} = (P/4 + 1) * RCOR * DOC$$

$$E_{RD} = (P/4 + 1) * .10 * .90$$

$$E_{RD} = 0.0225 * P + .09$$

In summary, the FCSC specific model(E_{FCSC}) for estimating programmer related effort in the Phase IV environment is as follows:

$$E_{FCSC-DAYS} = E_{TD} + E_{SW} + E_{DF} + E_{CL} + E_{ST} + E_{AT} + E_{RD}$$

$$E_{FCSC-DAYS} = P + 1.67 * STLOC * 10^{-3} + 3.3 * P + .75 * P + 1.069 * P + .15 + 1.45 * P - 1.4 * P * e^{-(P/20)} + .0225 * P + .09$$

$$E_{FCSC-DAYS} = 7.59 * P + 1.67 * STLOC * 10^{-3} - 1.4 * P * e^{-(P/20)} + 0.24$$

To convert to effort hours(E_{FCSC}), a multiplier of 8 hours per day was used to finally produce

$$E_{FCSC} = 60.72 * P + 13.36 * STLOC * 10^{-3} - 11.2 * P * e^{-P/20} + 1.92$$

Hahn and Stone or MITRE Model

Hahn and Stone represented the cost of conversion with the three component costs of program transfer(C_p), data transfer(C_D) and other costs(C_O)[Hahn and Stone, 1970]. The cost for transferring a program includes both costs of automatic(C_A) and manual(C_M) translation. Only the process for computing manual translation costs are further defined. The main element is a formula for calculating the total number of man-days required (MD_T). Several parameters must be defined before the MD_T formula may be used. Since no modifications are allowed during the conversion of Phase IV programs, $D_{F2} = 0$ and since there are no subprograms defined then $D_{F3} = 0$. The documentation status factor(D_{F1}) is one minus DOC of the FCSC model; in other words, since $DOC = .9$, $D_{F1} = .1$. For the COBOL translation environment, recoding is measured with $R_{BC} = 29$ and testing with $R_{BT} = 18.3$. The formula also requires a statement of the LOC to be manually transferred(I) and this may be defined as $.1 * STLOC$ for the Phase IV environment in which is used a 90% effective automatic translator. Therefore,

$$MD_T = I/R_{BC} + (D_{F1} * I/R_{BC}) + (D_{F2} * I/R_{BC}) + I/R_{BT} + (D_{F3} * I/R_{BT})$$

becomes

$$MD_T = \frac{.1*STLOC}{29} + \frac{.1*(.1*STLOC)}{29} + \frac{.1*STLOC}{18.3}$$

$$MD_T = \frac{4.9*STLOC}{530.7}$$

Using the FCSC estimate of 630 LOC per day for an automatic translator, the automatic translation time(AD_T) may be calculated with:

$$AD_T = \frac{.9*STLOC}{630}$$

Therefore, the total effort hours(E_{HS}) required for program conversion can be defined as:

$$E_{HS} = (MD_T + AD_T)*8$$

$$E_{HS} = \frac{8*4.9*STLOC}{530.7} + \frac{8*.9*STLOC}{630}$$

$$E_{HS} = .0853*STLOC$$

This formula must be supplemented with some factors to account for the C_D and C_O costs that are not further defined by Hahn and Stone. The additional effort estimates for test data generation and validation, file conversion and redocumentation were included. The FCSC estimates were used as a basis for defining these supplemental hours required. For test data generation, the FCSC estimate of 1 day or 8 hours per program(P) was used. Since Hahn and Stone specifically state that file or data conversion costs are typically small in comparison to program conversion costs, one-half of the FCSC estimate of 3.3 days per program was used. This results in 1.65 days or 13.2 hours per program. Since the FCSC estimate for

redocumentation seems too small, a factor of .5 days or 4 hours per program was selected. The total supplemental hours used were:

$$8*P + 13.2*P + 4*P = 25.2*P$$

Thus, E_{HS} finally becomes:

$$E_{HS} = .0853*STLOC + 25.2*P$$

Grim, Epler and Andrus Model

Grim, Epler and Andrus present a formula for computing the conversion programming cost in man-days(M)[Grim, Epler and Andrus, 1978]. This principal element of their model requires the definition of three parameters: the translator effectiveness(T) which is again assumed to be 90%, the documentation status (D), assumed to be .90, and the daily LOC conversion rate(R) of an average programmer set at 30 as suggested by the model for COBOL to COBOL translations. Therefore,

$$M = \frac{2*STLOC(1 - T)}{R*(1 + D)}$$

becomes

$$M = \frac{2*STLOC(1 - .90)}{30*(1 + .90)}$$

$$M = 0.00351*STLOC$$

It is assumed by the model that automatic conversion(A) is minimal and therefore not included. However, to provide a uniform basis for comparison, the FCSC estimate for automatic conversion was also used here:

$$A = \frac{.9*STLOC}{630}$$

$$A = .00143*STLOC$$

Therefore, the total effort hours(E_{GEA}) were computed as follows:

$$E_{GEA} = (0.00351*STLOC + .00143*STLOC)*8$$

$$E_{GEA} = .0395*STLOC$$

This formula was also adjusted to include relevant supplemental factors. Since data conversion costs using the model's suggested I/O unit cost approach can not be easily computed for Phase IV programs and to provide for uniformity, the same supplemental estimates used with the Hahn and Stone formulation were used here. The adjusted model is then:

$$E_{GEA} = .0395*STLOC + 25.2*P$$

Wolberg Model

Wolberg developed an effort estimation model for recoding based on nine very large projects which included time for all activities from planning to implementation[Wolberg, 1983]. This model produces person-month(E) estimates based on thousands of lines of code(L):

$$E = 7.14*L^{0.47}$$

For uniformity, the model was adjusted to estimate hours by multiplying by 173.2 which is Wolberg's estimate of person-hours per person-month. This results in the following estimate of hours(E_W):

$$E_W = 1237*L^{0.47}$$

To use the same basis of STLOC, the model became:

$$E_W = 1237*(STLOC/1000)^{0.47}$$

or

$$E_W = 48.12*STLOC^{0.47}$$

Since E_w results in estimates for the entire time spent by all the staff of a conversion project, an arbitrary factor of 50% was used to adjust the estimate to produce an estimate for programmer related tasks only. Thus, for comparison with other models the following is used to represent Wolberg's approach:

$$E_w = 24.06 * STLOC^{0.47}$$

Basili and Freburger Model

Basili and Freburger developed various models for new development efforts[Basili and Freburger, 1981]. One of the models uses the concept of developed lines(DL) which is defined to equal the number of new lines plus 20% of the reused lines. This model was considered to be possibly applicable in the conversion realm if reused lines are defined to be those converted by the automatic translator and new lines as those that require manual recoding. Since DL is defined in terms of thousands of LOC and a 90% effective translator is used:

$$DL = \frac{.1 * STLOC}{1000} + \frac{.20 * .9 * STLOC}{1000}$$

$$DL = 2.8 * STLOC * 10^{-4}$$

Basili and Freburger produced a linear fit using DL and generated the following model:

$$E = 1.46 * DL^{0.98}$$

where E is measured in person-months. This estimate was changed to hours by multiplying by 173.33 which is assumed in the model, to be the number of person-hours per person-month. Substituting the

expression for DL derived above, an estimate of effort hours(E_{BF}) can be calculated as follows:

$$E_{BF} = 1.46 * (2.8 * STLOC * 10^{-4})^{0.98} * 173.33$$

$$E_{BF} = .0835 * STLOC^{0.98}$$

Validation of Existing Models

The models defined in the previous section were used to predict effort hours which were then compared to the actuals by means of $R^2_{\text{Prediction}}$ (the percent variability in the data explained by the model), the average prediction error (average deviation in hours) and the average squared prediction error, which is comparable to the residual mean square (assumed to measure the average variance of the residuals from the fit) [Montgomery and Peck, 1982].

Measurement of Accuracy of Basic Models

Table 10 lists the effort estimation models that were used to predict effort hours for comparison with the actual hours. The validation/accuracy measures of the models are shown in Table 11. It was quite surprising to see that the Hahn and Stone model, which was suggested for use within Phase IV, exhibited the highest R^2 , the lowest average squared prediction error and an average prediction error with the lowest absolute value. The highly parameterized FCSC model had, surprisingly, the worst performance. This could be attributed to errors or incorrect assumptions made during the formulation of the Phase IV specific equation; however, a review of the process revealed no apparent problems.

Table 10. Summary of Conversion Effort Estimation Models.

| | |
|------------|--|
| E_{FCSC} | $=60.72*P+13.36*STLOC*10^{-3}-11.2*P*e^{-(P/20)}+1.92$ |
| E_{HS} | $=0.0853*STLOC+25.2*P$ |
| E_{GEA} | $=0.0395*STLOC+25.2*P$ |
| E_W | $=24.06*STLOC^{0.47}$ |
| E_{BF} | $=0.0835*STLOC^{0.98}$ |

Table 11. Validation/Accuracy Measures of Basic Models.

| Model | R^2_{Pred} | Average Squared Pred. Error | Average Pred. Error (Hours) |
|------------|--------------|-----------------------------|-----------------------------|
| E_{FCSC} | 0.440 | 729236 | 680.72 |
| E_{HS} | 0.841 | 207681 | 99.18 |
| E_{GEA} | 0.538 | 602174 | 607.30 |
| E_W | 0.671 | 428055 | -380.53 |
| E_{BF} | 0.685 | 408372 | 465.94 |

It is interesting to note that the Basili and Freburger model ranked second in terms of R^2 and the average squared prediction error. The concept of developed lines seems to have some applicability in the conversion area when code manually recoded is substituted for new code and code automatically translated is substituted for reused code. The Wolberg model was the only one that resulted in a negative average prediction error indicating that the predicted hours were generally higher than the actual hours. The

Grim, Epler and Andrus model had the second worst performance and since it resembles the structure of the Hahn and Stone model but with a smaller coefficient of STLOC it was not considered for further refinement and analysis. The FCSC model was also considered to be too inaccurate for this context with an R^2 of 0.44 and a large number of estimated parameters so it was not studied further.

To present a pictorial view of the behavior of the basic effort estimation models, a hypothetical system of 8 programs with a varying number of lines of code was used for making estimates with the models. Figure 3 contains the overlaid plot of the five models. Each is represented with the first character of the model's name. Notice that the FCSC(labeled F in the plot) model and the Grim, Epler and Andrus(labeled G) model produce estimates that are much lower than those of the other models.

Analysis of Refined Models

Only the Hahn and Stone, Wolberg, and Basili and Freburger models were considered for further refinement and analysis as discussed above. Though the average squared prediction error for the Hahn and Stone model was significantly less than that of the other models, it was still somewhat large in magnitude. Attempts were made to reduce this error and increase R^2 for all three models.

First, the Wolberg model results were reviewed. Since, on the average, the model estimates were higher than the actuals, it was decided to reduce the arbitrary percentage of programmer related time

SAS

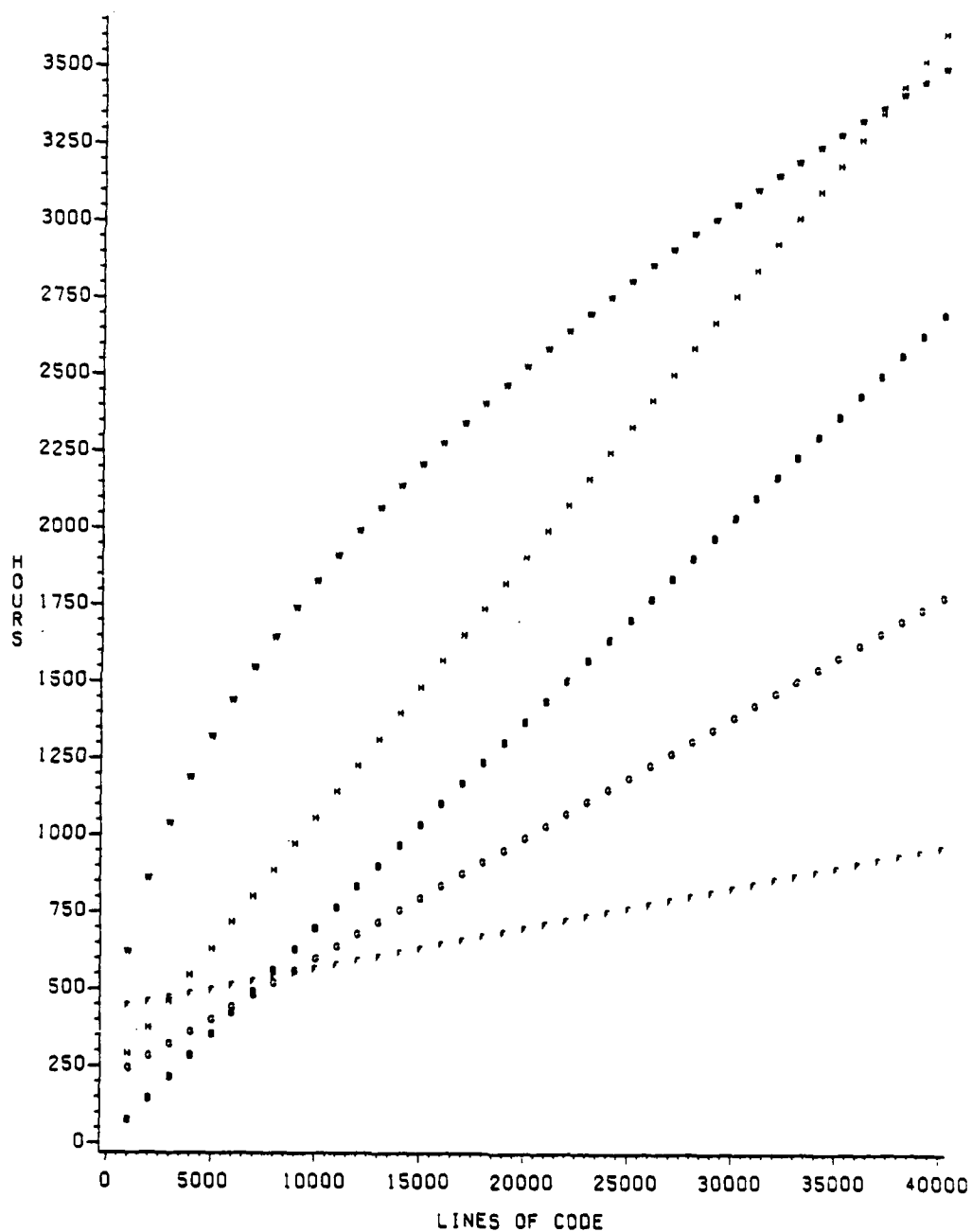


Figure 3. Plots of Estimates of Basic Models for a Small System

from the 50%(reduction of time) of the original model which included time for the entire project, from planning to implementation. Three refinements, with 30, 35, and 40 percent, were attempted. The model which assumed 40% of the overall time, for programmer related functions, demonstrated the best results:

$$\text{Revised } E_w \text{ Model: } 19.25 * STLOC^{0.47}$$

| R^2 | Avg.Squared Pred. Error | Avg. Pred Error |
|-------|-------------------------|-----------------|
| ----- | ----- | ----- |
| 0.742 | 335932 | -59.42 |

Though R^2 was increased and the average prediction error was decreased, the average squared prediction error is still relatively high. The data used for development of the original Wolberg model was very different from that of Phase IV so better results than these would be unlikely.

The Basili and Freburger model which performed remarkably well in the basic analysis was adjusted by modifying the equation for calculating the number of developed lines. The original equation called for summing the new code plus 20% of the reused code. This produced the model used in the comparisons of the previous section. It was decided to try various percentage factors to calculate the overhead associated with reused code, or in this case, with code translated by the automatic translator. The original performance measures showed that the model estimates were normally lower and thus percentages higher than 20% were required. Adjustment percentages of 40, 35, 30, and 25 were used for comparison. The 30% overhead factor produced the best results as follows:

Revised E_{BF} Model: $0.1097 \cdot STLOC^{0.98}$

| R^2 | Avg. Squared Pred. Error | Avg. Pred. Error |
|-------|--------------------------|------------------|
| 0.823 | 229995 | 227.54 |

These results were better than expected with R^2 increasing to a very adequate level and the average prediction error decreasing considerably. However, the fairly large average squared prediction error indicates the existence of some large residuals. The predicted hours are much better estimates of the actuals than the average of the actual hours; i.e., the total sum of squares about the mean is much larger than the average squared prediction error. This results in an apparently commendable R^2 .

The Hahn and Stone model had the best overall performance in the initial analysis. Even after improving the results of the next best models, Wolberg and Basili and Freburger, the Hahn and Stone model continued to exhibit better performance. One parameter that this model uses that is not present in the other two models is that of the documentation status. It was decided to test the sensitivity of the model results to the documentation status used by Hahn and Stone (see Appendix A) by changing the status from .1 (very good documentation) to .25 (good documentation). The coefficient of $STLOC$ changed from 0.0853 to 0.0896. The second term of the model ($25.2 \cdot P$), developed by using FCSC model criteria, was not initially modified. The results of this revised model indicated a decrease in R^2 and an increase in the average squared prediction error. Though the changes were not great, the results demonstrated that a documentation status of .1 was most appropriate.

Another element that makes the Hahn and Stone model different from the Basili and Freburger and Wolberg models is the presence of a term which attempts to capture the effort involved in transferring data and in other activities. With the coefficient of STLOC set at its original value of 0.0853, several different coefficients of P(number of programs) were used to determine their impact on the model's performance. All of the coefficients greater than the original value of 25.2 degraded the model's performance while only a couple of lower valued coefficients provided a slight improvement. The model with the best R^2 and the lowest average squared prediction error was the following:

Revised E_{HS} Model: $0.0853 \cdot STLOC + 20 \cdot P$

| R^2 | Avg. Squared Pred. Error | Avg. Pred. Error |
|-------|--------------------------|------------------|
| 0.845 | 201816 | 136.36 |

Though providing a slight improvement in results, this model ranks about equal with the original model because there is an accompanied increase in the average prediction error.

Development of Models With Regression Analysis

The availability of Phase IV system level effort data presented an excellent opportunity to develop models specific to the Air Force environment. One model of exponential form, similar to that of Wolberg and Basili and Freburger, was developed. A second model of additive form, such as that of the Hahn and Stone model, was also constructed.

Exponential Form Effort Model

The first regression model developed produced an R^2 of 0.844 which is about the same as that of the Hahn and Stone model. The resulting parameters formed the following equation for predicting effort hours(E_{AFX}):

$$E_{AFX} = 5.55 * STLOC^{0.591}$$

Since logs of the actual hours and the lines of code were used to build the model, it is difficult to compare the resulting mean square error(MSE) with previous average squared prediction errors. However the model's F value of 97.69 is several times the size of the critical F value of 4.14 indicating model adequacy and predictive value[Draper and Smith, 1966]. The residual plot demonstrated an adequate scatter of the points around zero thus contributing to the assessment of an adequate model. The prediction error sum of squares or PRESS statistic, as discussed in the previous chapter, was used for partial validation of the model as follows:

$$R^2_{\text{Prediction}} = 1 - \frac{\text{PRESS}}{\text{Corrected Sum of Squares}}$$

$$R^2_{\text{Prediction}} = 1 - \frac{2.86663}{14.69262} = 0.805$$

This result indicates that this model could be expected to explain about 80 percent of the variability in predicting new observations, as compared to the 84 percent of the variability in the original data explained by the least squares fit. The "loss" in R^2 for prediction is very slight indicating model adequacy.

Additive Form Effort Model

The additive model was developed with starting lines of code(STLOC) as well as number of programs(P). The resulting equation for predicting effort hours(E_{AFA}) was as follows:

$$E_{AFA} = 309 + 0.0390 \cdot STLOC + 67.76 \cdot P$$

This model produced the highest R^2 (0.928) of the effort estimation analysis indicating that 92.8% of the variability of the actual hours is explained by the model. The model's F value of 109.36 was even higher than that of the exponential form model. The mean square error(MSE), 110520, was lower than the average squared prediction error of all of the existing models studied above. The PRESS statistic was computed again and used with the corrected sum of squares to calculate a prediction R^2 for partial model validation:

$$R^2_{\text{Prediction}} = 1 - \frac{6162413}{26051370} = 0.763$$

This model could be expected to explain about 76% of the variability in predicting new observations(compared to 80% of the exponential form model), as compared to the 92.8% in the original data explained by the least squares fit. The "loss" in R^2 for prediction is greater than that of the exponential model; however, model adequacy is still upheld. Though the residual plot demonstrates an adequate spread around zero, a couple of "distant" points reveal the variance which contributes to the lower R^2 for prediction.

Final Comparison of Models

Using the same hypothetical case used earlier, the refined Wolberg, Basili and Freburger, and Hahn and Stone models were plotted along with the two regression models. Figure 4 contains the plots of the five models with the exponential form model represented by "1" and the additive form model represented by "2" while the other models are again represented by the first letter of the model's name. Notice that all the models operate within a much closer framework. The additive form model (labeled 2) depicted the highest R^2 when regressed against the Phase IV data.

In order to provide a further evaluation of the two regression models, the models were assumed to be developed independent of the data and were both used to predict hours for comparison with the actuals, as was done for the validation of existing models. The exponential(E_{AFX}) model produced the following results:

| R^2_{Pred} | Avg. Squared Pred. Error | Avg. Pred. Error |
|--------------|--------------------------|------------------|
| 0.791 | 271700 | 69.81 |

The performance of the additive(E_{AFA}) regression model appeared to be somewhat better as indicated by the measurements:

| R^2_{Pred} | Avg. Squared Pred. Error | Avg. Pred. Error |
|--------------|--------------------------|------------------|
| 0.928 | 93943 | -0.460 |

The average prediction error is very close to zero which means that the model seems to produce approximately unbiased predictions [Montgomery and Peck, 1982]. The additive model appears to be a better predictor of effort hours. A numerically simpler

SAS

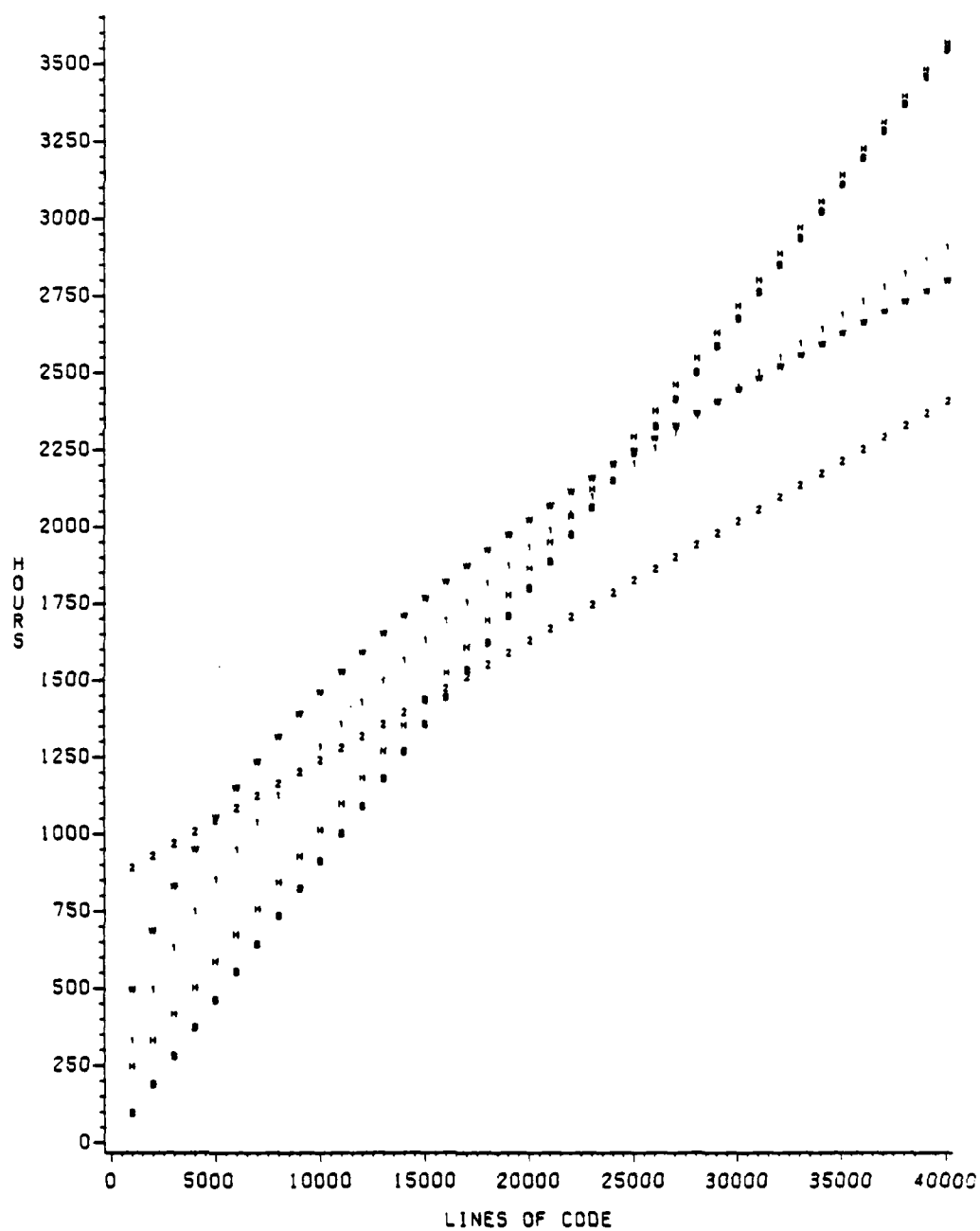


Figure 4. Plot of Refined Models and Regression Developed Models

version of the additive model was validated with the objective of producing a model with greater usability. The following simpler model:

$$E_{AFA} = 300 + .039 \cdot STLOC + 70 \cdot P$$

demonstrated the following performance:

| R^2_{Pred} | Avg.Squared Pred. Error | Avg.Pred. Error |
|--------------|-------------------------|-----------------|
| 0.928 | 94422 | -7.48 |

The accuracy of the simplified model is only slightly different from that of the original additive model. Therefore, it is well suited for this environment. Further validation of these models could be done as future work when more data is available.

Organizational Impact Model

One last step in the analysis evaluated the impact of the organizational factor. The additive form model was extended to include a categorical variable(MAJCOM) and the GLM SAS procedure, with MAJCOM defined as a CLASS variable, was run to develop the model and measure its adequacy. Table 12 demonstrates the results of the regression analysis including the model coefficients. Notice that this model produced the highest R^2 (0.968) of all the models investigated. The model F value is very large and several times more than adequate. The mean square error is, by far, the lowest value encountered in the analysis. All the parameters are significant at least at the 0.0004 level. The coefficients of STLOC and P changed very slightly from the basic additive form of the previous section.

Table 12. Organizational Impact Effort Model

DEPENDENT VARIABLE: HOURS

| SOURCE | DF | SUM OF SQUARES | MEAN SQUARE |
|-----------------|----|----------------|-------------|
| MODEL | 03 | 25210438 | 8403479 |
| ERROR | 16 | 840932 | 52558 |
| CORRECTED TOTAL | 19 | 26051370 | |

| MODEL F | PR > F | R-SQUARE | C.V. | HOURS MEAN |
|---------|--------|----------|------|------------|
| 160 | 0.0001 | 0.968 | 18.7 | 1225.7 |

| SOURCE | DF | TYPE III SS | F VALUE | PR > F |
|--------|----|-------------|---------|--------|
| STLOC | 1 | 1053990 | 20.05 | 0.0004 |
| P | 1 | 1087373 | 20.69 | 0.0003 |
| MAJCOM | 1 | 1037910 | 19.75 | 0.0004 |

| PARAMETER | ESTIMATE |
|-----------|----------|
| INTERCEPT | 88.8398 |
| STLOC | 0.0399 |
| P | 61.1284 |
| MAJCOM-O | 468.1244 |

However, the organizational influence in the model became clearly visible when a coefficient of 468.1243 appeared for the case of MAJCOM being AFSDC. This difference could possibly be attributed to the more complex and larger systems converted by AFSDC or because of insufficient attention having been given to conversion planning and preparation.

A partial validation of the model, with the PRESS statistic, revealed:

$$R_{Pred}^2 = 1 - \frac{\text{PRESS}}{\text{Corrected Sum of Squares}} = 1 - \frac{3525349}{26051370} = 0.865$$

This value of R^2 reflects a slight loss which indicates that the linear fit explains a greater percentage of the original data;

however, the model is quite adequate. The residual plot showed a good scatter with only one point somewhat distant from the rest. Another adequacy test was to assume the model's independence from the data and determine predicted hour values based on the data. This produced the following:

| R^2_{Pred} | Avg.Squared Pred. Error | Avg. Pred. Error |
|---------------------|-------------------------|------------------|
| 0.968 | 42047 | -0.3061 |

The high accuracy of the model reveals that the organizational factor is a significant element of effort estimation.

CHAPTER VII

SUMMARY AND RECOMMENDATIONS

Introduction

Overview of Work Accomplished

The work performed in this research parallels the objectives and procedures originally formulated. A two-phased skeletal methodology was initially conceived for the study of software conversion data. Each of the phases, productivity analysis and effort estimation analysis, was subjected to close scrutiny with a keen eye on new software development research, since software conversion research was found lacking. Specific sets of procedures and statistical analyses were formulated for each of the phases of the study. This gave life to a methodology for conversion data analysis which permitted a careful study of conversion productivity and effort estimation by means of a case study of Air Force Phase IV conversion data from organizations in different locations working on unique and independent systems.

Program and programmer attributes became dependent variables as their impact on programmer productivity was analyzed. Models for explaining productivity were constructed and the impact of organization was also considered. Existing applicable software conversion effort estimation models were validated for accuracy using the Phase IV data and environment specific regression models were constructed.

Significance of Research Outcomes

This research benefits the computer community since a methodology for conversion data analysis was formulated. Also, an analysis of conversion productivity and a study of conversion effort estimation models were lacking. Most of the work accomplished fits within the general framework of measurement studies. One of the goals of the STARS program in the measurement area is to encourage the development and refinement of measures and models[Dunham and Kruesi, 1983]. Coincident with this goal is a suggestion that this activity be carried out within the context of on-going software projects. This research not only fulfilled this goal of STARS, but it was also conducted within the context of the on-going Phase IV software conversion project. A conversion programmer productivity case study had not been previously conducted on this scale. This research generated a productivity model and a cost or effort estimation model for the Phase IV environment which may be applicable to other environments with small systems written in a high level language and being converted to a highly compatible high level language and operating within a multi-location government organizational structure. A validation of these models can be accomplished as future work when more data becomes available.

Summary of Methodology Formulated

The methodology for the analysis of conversion programmer productivity and effort estimation emerged from the selection of appropriate statistical techniques used throughout this study. One

of the primary techniques employed was that of regression analysis which is one of the most widely used techniques for analyzing multifactor data[Montgomery and Peck, 1982]. The methodology is naturally based on two separate and distinct phases which encompass the analysis associated with each of the two aspects of software conversion chosen for this research.

The first phase of the methodology is the study of conversion productivity. The five steps which provided the framework for this phase were:

- 1) collect and prepare raw data on programmers and programs,
- 2) perform preliminary analysis of data,
- 3) construct appropriate file of programmer and program data elements,
- 4) build productivity model for exploratory analysis, and
- 5) perform model validation.

The second phase of the methodology is the study of conversion effort estimation. This phase also has five steps and parallels the first phase:

- 1) collect appropriate effort data,
- 2) select conversion estimation models,
- 3) produce effort estimates and compare to actuals,
- 4) build effort estimation models for specific environment, and
- 5) perform model validation.

Productivity Methodology

The first step of the methodology for studying conversion productivity was the collection and preparation of raw data. One set of data required was that of programmer resumes and the other set consisted of program information. Some programmer qualitative attribute data was provided in a form unsuitable for data entry and manipulation. Therefore, an encoding scheme was devised and applied to the data. A data entry process was selected and the data was keyed into two separate data files.

The second step of the methodology was an elementary analysis of the separate data files. Programmer qualitative variables were tested to determine pair-wise associations. An examination of the correlation between the programmer quantitative variables was also conducted. A productivity measure(LOCPERHR) and a program difficulty measure were calculated from the program information data. Tabular summaries were prepared and a variety of elementary statistics were calculated.

The third step of the methodology was the formation of a new file which included all the programs converted by only one programmer. The attributes of the programmer from the programmer file were also added to each record of this new file, called the individual productivity file. This file provided the basis for continuing the analysis.

The fourth step of the methodology was that of variable selection and model building with the individual productivity file.

Multicollinearity, duplication of a variable, inadequacy of data, and simplicity were used as criteria for candidate variable selection. The selected qualitative and quantitative variables became independent variables with LOCPERHR as the dependent variable. Regression analysis was initiated with this candidate set of variables. Variables found to be insignificant in the model were eliminated. The adequacy of the final model was checked by means of R^2 , residual analysis, and reasonableness of the resulting equation. The resulting model parameters and the significance of their impact on productivity were discussed. Also, the impact of the variable of organization was studied by adding it to the model and reinitiating the regression analysis.

The fifth step of the productivity methodology was that of model validation. This typically requires the collection of new data. Since no additional data from the Phase IV program was presently available, a secondary form of validation, data splitting, was used. The prediction error sum of squares(PRESS) statistic, a form of data splitting, was determined and applied to the resulting model.

Effort Estimation Methodology

The first step of the methodology for studying conversion effort estimation is the collection of appropriate effort data. System level effort data which included effort hours, lines of code, and number of programs for each system was collected. A designator for the organization converting each system was also provided.

The second step of the methodology was the selection of conversion effort estimation models of interest. After a review of the literature, existing significant models were chosen. These models were studied individually and then represented by an estimation equation reflecting the characteristics of each model in a form suitable for using the collected data.

The third step of the methodology was the application of the data to the models to produce estimates for comparison. The average hour deviation(prediction error) and R^2 were used to evaluate the accuracy of estimated hours as compared to the actual hours. The three best models were selected for further study. The three equations were fine tuned in an attempt to improve their performance when compared to the actuals.

The fourth step of the methodology was the construction of effort estimation regression models based on the collected data. Two environment specific models were constructed. One model was built using an exponential form and the other an additive form. The impact of organization was studied by including it as a variable in the additive form model and reinitiating the regression analysis.

The fifth step of the effort estimation methodology was the validation of the models. Since no new Phase IV data was presently available, the prediction error sum of squares(PRESS) statistic, a form of data splitting, was used for model validation.

Summary of Productivity Analysis

The regression analysis performed on the productivity data resulted in the formulation of a model for explaining the productivity of programmers within the context of program attributes. The results of the productivity analysis are true for the sample data of the Phase IV environment but no conclusions may be drawn about other conversion environments.

The statistical analysis revealed that there is a slight increase in productivity as the starting lines of code increase. This parallels the results of Paulsen who found the same relationship in the development of products with a high level of reused code[Paulsen 1981]. Only conversions like Phase IV with a high percentage of code automatically translated may possibly experience this phenomenon. An explanation for this behavior may be that the fixed overhead effort is dominant when most of the code is translated automatically. This implies that as the numerator of STLOC increases, the denominator increases only slightly and thus the quotient of LOC/PERHR increases. This phenomenon may not extend to Phase IV programs greater than 5000 lines which are outside the range of the case study sample and it definitely does not extend to Phase IV conversions that are not COBOL-68 to COBOL-74. As the complexity or difficulty rating of a program increases there is an accompanying decrease in productivity, as expected.

An interesting manifestation, supporting a discovery by Oliver, was exhibited by the rating of a programmer's knowledge of a

program[Oliver, 1978]. Oliver stated that programmers converting their own programs may not resist the temptation to "improve" the program they convert. This thesis indicates that productivity decreased somewhat as the individual's knowledge of a program increased; therefore, knowledge of a program seemed to be unfavorable. This counter-intuitive phenomenon may not apply to other conversion environments and the result may change as the Phase IV conversion progresses. The finding suggests that programmers with greater knowledge of a program tend to perform unauthorized modifications or enhancements during the conversion process either for personal reasons or as directed by local management.

It is interesting to note that programmers who classified their experience as primarily of the development type exhibited higher productivity than maintenance type programmers or programmers with both development and maintenance experience inferring that development type programmers had the greatest depth of experience. This may not be true in private industry or other environments. The data shows, what may be common in many organizations, that the inexperienced programmers typically start out in maintenance. The programmer type variable was the only categorical(qualitative) variable to remain in the final LOCPERHR model. The influence of academic degree and major were found to be insignificant in the initial model formulation.

The results indicate that, within the Phase IV environment, experience with the source language(COBOL-68) is more important than experience with the target language(COBOL-74). In fact, the sample

data produced a model for productivity which completely eliminated COBOL-74 experience. General programming experience also had a positive impact on productivity paralleling other studies which indicate that productivity increases with experience.

The JCL experience variable revealed a peculiarity in the data which probably only exists in this environment. In most cases the JCL experience included primarily academic experience for junior programmers or programmer trainees. This was discovered during the preliminary analysis showing that JCL experience for intermediate and senior programmers in Phase IV was typically nil since there is no JCL for the B3500 environment programs. Therefore, the JCL experience in the data is almost all for entry level programmers and thus produces a negative impact on productivity. JCL experience is really more of an indicator for an entry-level programmer and nothing else.

When a programmer's organization entered the regression analysis, the results indicated support of the findings of Lawrence[1981] and Jeffery and Lawrence[1979] which state that the organization has an impact. Even in the context of well-defined procedures within the Air Force Phase IV Program, one organization demonstrated a productivity which was on the average about 150 lines of code per hour higher than that of another organization. Upon entering the organization variable in the final model, the programmer type and COBOL-68 experience variables were found to be insignificant. This indicated that the organization variable

replaced the combined effect of these two variables. The results point out that the productivity of organizations varies even within the context of the same industry as discovered by Lawrence.

The alternate productivity model of "cost units" or hundreds of lines of code per hour and the alternate model of the log of lines of code per hour both resulted in a similar set of explanatory or regressor variables. The most significant difference between these models and the basic or primary lines of code per hour model was the presence of the variable MAJOR which replaced the general programming experience variable. Both alternate models indicated that, within the Phase IV environment, programmers with computer science, data processing-MIS, or mathematics/engineering education were more productive than programmers in the category of other academic majors.

Summary of Effort Estimation Analysis

Five effort estimation models compatible with the software conversion arena were validated for measurement accuracy. It was surprising that the Hahn and Stone model, suggested for use by the AFASPO in Phase IV, exhibited the best performance. The Basili and Frebarger model, based on the concept of "developed lines of code", demonstrated its applicability to software conversion with acceptable performance. Though the Wolberg model was based on hour data for entire projects, a model assuming 50% of the effort for programmer related activities revealed fairly adequate results. The new FCSC model manifested low accuracy in comparison to the other models. The

correct usage of the FCSC model was insured by double checking its application. It is apparent that the FCSC model requires some additional study for environments of the Phase IV variety: high level to high level language conversions, good documentation and fairly compatible source and target environments. Though only one FCSC conversion class was examined in this research, the results indicate that a recheck of the other model classes may be in order.

The Hahn and Stone, Wolberg, and Basili and Freburger models were subjected to various tuning modifications aimed at improving their performance, within the Phase IV environment. The Wolberg model never rose above an R^2 of 0.74 while the Basili and Freburger model reported an R^2 of 0.823 when a 30% overhead factor for reused code (code translated automatically) was applied. The Hahn and Stone model demonstrated its best performance with an R^2 of 0.845 when a coefficient of 20 hours was used with $P(\text{number of programs})$.

Two regression models for the Phase IV environment were built using the data provided. The first model, of exponential form, resulted in an R^2 of 0.844 and exhibited traits of model adequacy. However, the additive form model which used both STLOC(starting lines of code) and $P(\text{number of programs})$ demonstrated an R^2 of 0.928. The R^2 for prediction for the additive model manifested about a 16% "loss", thus about 76% of the variability in new observations could be explained with the model. Suspecting an organizational factor of significance, another additive form model was developed using MAJCOM as a CLASS variable in the SAS GLM procedure. The resulting equation

exhibited an R^2 of 0.968 and showed that one organization, the AFSDC, added almost 500 more hours to the predicted effort value. This again indicated a significant organizational impact. This large difference in the organizations could originate from the possibly larger and more complex systems converted by the AFSDC or from insufficient pre-conversion preparation.

Management Considerations

It is imperative that further studies in this area be conducted to increase the community's understanding of the subject of conversion and to improve management's awareness of problems and opportunities. The foundation for these efforts is a data base supported by an effective data collection process. A discussion of data related problems experienced during this research and suggested enhancements of the data collection process as well as suggestions for selecting personnel for conversion projects are included in this section.

Data Collection Forms

Repeating the earlier discussion of the forms and the encoding required for the data is not appropriate at this point. However, some general comments are in order. Open-ended questions, such as the Formal Training question of the programmer's form, should be avoided since the variety of possible responses is enormous and an adequate analysis of the question will be difficult. Questions with complex conditionals, such as, "If the majority of programs you shall

be transitioning are not COBOL, then what type of system are they?", may cause some confusion. Direct and simple questions should be the rule and for the most part, this was true of Phase IV questions. The program information form could have included more questions requesting additional descriptions of the program, such as, input and output data, detailing numbers, sizes, types, etc. At the far end of the spectrum are the conversion data collection forms designed by the Data & Analysis Center for Software(DACS)[1981]. Though these forms have been available for about three years, they have not been used extensively because of the extreme number of details requested. The DACS should be commended, however, for its efforts to collect conversion data to establish a data base for use by the community.

Data Submission Procedures

If the forms are well designed but they are improperly completed, reliable analysis with adequate results will be impossible. A good form ~~must~~ be supported by better instructions and definitions for completion. Management of all levels must insure that all forms are well understood and properly completed. Some of the organizational differences detected in this research may have been due to different interpretations of the data collection forms and submission process. Many forms for programs converted were submitted prior to their being redocumented thus requiring a reporting of zero time for this activity. This time or effort category was subsequently deleted from the productivity analysis conducted.

Quality control of the program information forms is a must. This is required, not at the AFASPO which is the collection repository, but at all of the conversion centers responsible for completing these forms. The lack of quality control is evidenced by the forms of one center that typically volunteered an overall hour total for the program which included the programmer's knowledge code in the count of total hours.

Controlling the Process

There seemed to be an apparent lack of directive power in regards to the data collection process. The AFASPO had to practice expert appeasement skills as many organizations quibbled with the conversion and data collection process. At least one organization was permitted to forgo the completion of program information forms. Other organizations showed their disagreement in the little attention placed on correct completion of data forms. The advantages, to the entire Air Force, of having sound productivity and effort data for analysis are apparently not clearly seen by all. This is a management problem which could be present within any large corporation. Better marketing of ideas and processes, such as data collection, is a must for software engineers involved in conversion, as well as in any other area.

Personnel Selection Considerations

One of the significant management tasks required prior to a conversion is the staffing of the project team. This thesis research indicates some criteria that may be used by a Phase IV manager to select the best personnel for the conversion effort. Programmers with little or no knowledge of the programs to be converted are preferred since these individuals will be least prone to make modifications or enhancements during the conversion process thus maintaining the required level of reduced risk. Maintenance type programmers should only be used if they are well experienced. Development type programmers with a few years of experience are primary candidates for selection. Programmers with a foundation in computer science, data processing-MIS, or mathematics/engineering are preferred. Knowledge of both the source and target languages is important; however, experience with the source language is more significant than experience with the target language. Since the organizational element is a factor to be considered, an individual with previous experience in a well managed organization with a good technical reputation is a better candidate for selection.

Future Research Possibilities

The research conducted serves ideally as a springboard for many additional investigations. Most related work in the new development environment may be repeated within the conversion context. This thesis research parallels studies in the development environment and

the same procedural steps should be repeated when additional data becomes available. The Phase IV conversion is due for completion in late 1985 at which time a complete analysis would be in order.

The impact of the organization on the conversion process and on the programmers' productivity should be investigated further. A definition of specific organizational traits should be developed to assist in the identification of productivity or overall effort differences. The list of traits should include management techniques, development methodologies, programmer tools, military vs civilian categorizations, etc.

The productivity of a group of programmers involved in the conversion of one program should be studied. Methods for defining group personnel characteristics, such as education and experience, should be investigated. An examination of the personnel characteristics in relation to the program details and overall productivity is of interest.

The FCSC model formulation used in the effort estimation analysis demonstrated low accuracy. An investigation of the model details to determine reasons for this performance is definitely in order. The compatible versus non-compatible(source and target) environments require the employment of different FCSC model formulations. These and other parameter choices should be investigated.

This research involved only COBOL to COBOL program translations. When the few FORTRAN and Assembler to COBOL program translations do

take place, a new analysis of the associated effort and productivity is a must. Data from other Air Force or non-government projects can be used to check the COBOL to COBOL translation results or to provide a source of information for other types of language conversions.

The effort estimation work can be expanded to cover the entire spectrum of the conversion project. However, data for this analysis may be difficult to come by. It may still be possible to collect historical conversion planning and preparation hours of all Phase IV organizations and initiate procedures for the capture of all post-conversion effort hours. This data would lead to an investigation of the applicability of the time-based effort estimation Rayleigh model.

The software conversion realm of the computer field has not received sufficient attention from researchers and it is becoming increasingly important that this area not be overlooked. Computer science researchers are primarily responsible to the computer community for studying all aspects of the field to determine ways of improving the availability and usability of the computer resource. The millions of dollars being spent for conversions every year are an indication of the significance of conversions in the industry. Much work is necessary to insure that financial resources are being effectively expended for conversions. The areas addressed by this research, programmer productivity and effort estimation, and the recommended research above are but a small part of the overall effort required.

REFERENCES

- AHTOLA, O. AND WILDT, A.R. 1978. Analysis of Covariance. Sage Publications, Inc., Beverly Hills, CA.
- AIR FORCE AUTOMATED SYSTEMS PROJECT OFFICE. 1982a. "Phase IV Development Center Software Transition Guidance Package." (Feb.). AFASPO/PGYW, Gunter AFS, AL.
- AIR FORCE AUTOMATED SYSTEMS PROJECT OFFICE. 1982b. "Phase IV Data Project Plan." (June). AFASPO/PGC, Gunter AFS, AL.
- AIR FORCE AUTOMATED SYSTEMS PROJECT OFFICE. 1983. "Program Information Form and Programmers' Resumes." (Letter, Apr.). AFASPO/PGY, Gunter AFS, AL.
- AIR FORCE AUTOMATED SYSTEMS PROJECT OFFICE. Undated. "Analyze Contractor Conversion Techniques and Programming Methodology." Report No. STC 404. AFASPO/PGY, Gunter AFS, AL.
- AIR FORCE DATA SYSTEMS DESIGN CENTER. 1982. "AFDSDC In-House Software Transition Plan." AFDSDC/DMBF, Gunter AFS, AL.
- AITKIN, M.A. 1974. "Simultaneous Inference and the Choice of Variable Subsets in Multiple Regression." Technometrics 16, 2(May), 221-227.
- ARON, J. D. 1969. "Estimating Resources for Large Programming Systems." Report on a Conference Sponsored by NATO Science Committee. (Oct.). Rome, Italy. Also in Tutorial: Software Cost Estimation and Life Cycle Control, L.H. Putnam, Ed. IEEE Computer Society Press, Los Alamitos, CA, 1980.
- BAILEY, J.W. AND BASILI, V.R. 1981. "A Meta-Model for Software Development Resource Expenditures." In Proc. 5th Int. Conf. on Software Engineering (Mar.), IEEE Computer Society Press, Los Alamitos, CA, pp. 107-116.
- BASIL, V.R. AND FREBURGER, K. 1981. "Programming Measurement and Estimation in the Software Engineering Laboratory." Journal of Systems and Software 2, 2(June), 47-57.
- BOEHM, B.W. 1981. Software Engineering Economics. Prentice-Hall, Englewood Cliffs, NJ.
- BOOCH, G. 1983. Software Engineering with Ada. Prentice-Hall, Englewood Cliffs, NJ.
- BROOKS, F. 1975. Mythical Man-Month. Addison-Wesley, Reading, Mass.

- CHAPIN, N. 1981. "Productivity in Software Maintenance." In Proc. AFIPS 1981 Nat. Computer Conf... vol. 50., AFIPS Press, Arlington, Va, pp. 349-352.
- CHRYSLER, E. 1978. "Some Basic Determinants of Computer Programming Productivity." Commun. ACM 21, 6(June), 472-483.
- COLLICA, J., SKALL, M. AND BOLOTSKY, G. 1980. "Conversion of Federal ADP systems: A Tutorial." (Aug.). National Bureau of Standards, Publication No. 500-62.
- COX, D.R. AND SNELL, E.J. 1974. "The Choice of Variables in Observational Studies." Applied Statistics 23, 1, 51-59.
- CROSSMAN, T.D. 1979. "Taking the Measure of Programmer Productivity." Datamation 25, 5(May), 144-147.
- DATA & ANALYSIS CENTER FOR SOFTWARE. 1981. "DACS Conversion Data Collection Forms." (June). DACS, RADC/ISISI, Griffiss AFB, NY.
- DATAPRO. 1983. DATAPRO Directory of Software. Datapro Research Corporation, Delran, NJ.
- DeMARCO, T. 1982. Controlling Software Projects: Management, Measurements & Estimation. Yourdon Press, New York, NY.
- DEPARTMENT OF DEFENSE. 1983. Software Technology for Adaptable, Reliable Systems(STARS) Program Strategy. (Mar. 15). DoD Publications.
- DRAPER, N.R. AND SMITH, H. 1966. Applied Regression Analysis. John Wiley & Sons, Inc., New York, NY.
- DUNHAM, J. AND KRUESI, E. 1983. "The Measurement Task Area." Computer 16, 11(Nov.), 47-55.
- FEDERAL CONVERSION SUPPORT CENTER. 1981a. "Review and Analysis of Conversion Cost-Estimating Techniques." Report No. GSA/FCSC-81/001. (Apr.). FCSC, Falls Church, VA.
- FEDERAL CONVERSION SUPPORT CENTER. 1981b. "Conversion Contracting Techniques Associated With Procurement of a Replacement ADP Hardware System." Report No. GSA/FCSC-81/003. (Sep.). FCSC, Falls Church, VA.
- FEDERAL CONVERSION SUPPORT CENTER. 1982a. "Federal Conversion Support Center Conversion Cost Model(Version 2)." Report No. OSD/FCSC-82/001. (June). FCSC, Falls Church, VA.
- FEDERAL CONVERSION SUPPORT CENTER. 1982b. "Conversion Work Package." Report No. OSD/FCSC-82/002. (July). FCSC, Falls Church, VA.

- FEDERAL CONVERSION SUPPORT CENTER. 1982c. "FCSC Conversion Tools Survey." Report No. OSD/FCSC-83-001. (Oct.). FCSC, Falls Church, VA.
- FEDERAL CONVERSION SUPPORT CENTER. 1983a. "Conversion Plan Outline." Report No. OSD/FCSC-83-002. (Jan.). FCSC, Falls Church, VA.
- FEDERAL CONVERSION SUPPORT CENTER. 1983b. "Software Conversion Lessons Learned." Report No. OSD/FCSC-83/003. (Jan.). FCSC, Falls Church, VA.
- FERNANDEZ, J.D. 1982. "Software Engineering Economics." Technical Report No. TAMUDCS-82-004-R, (May), Department of Computer Science, Texas A&M University.
- FERNANDEZ, J.D. AND SHEPPARD, S.V. 1984. "Software Conversions to Ada Require Unique Planning." Submitted to Defense Management Journal for publication in 1984.
- FREUND, R.J. AND LITTELL, R.C. 1981. SAS For Linear Models: A Guide to the ANOVA and GLM Procedures. SAS Institute Inc., Cary, NC.
- FRY, J.P., LOWENTHAL, E., SHOSHANI, A., BIRSS, E., LUM, V., SU, S., DRESSEN, P., MARION, R., SWARTWOUT, D., GOGUEN, N., NAVATHE, S., TAYLOR, R., KAPLAN, M., SCHINDLER, S., AND YORMACK, B. 1978. "An Assessment of the Technology for Data- and Program-related Conversion." In Proc. AFIPS 1978 Nat. Computer Conf., vol. 47. AFIPS Press, Arlington, Va, pp. 887-907.
- GENERAL ACCOUNTING OFFICE. 1977. "Millions in Savings Possible in Converting Programs from One Computer to Another." GAO Report FGMSD-77-34, (Sep. 15).
- GRIM, G.D., EPLER, E.D., AND ANDRUS, W.L. 1978. "Estimating the Cost of Conversion." In Proc. of Computer Related Information Systems Symposium. Sponsored by the U.S. Air Force Academy, (Jan. 25-27), Colorado Springs, CO.
- HAHN, W. AND STONE, J. 1970. "Software Transfer Cost Estimation Technique." (July). MITRE Corporation, Bedford, Mass.
- IBM. 1981. IBM PC Disk Operating System. IBM Corp., Boca Raton, FL.
- ITT RESEARCH INSTITUTE. 1979. Quantitative Software Models. Report for the Data and Analysis Center for Software of Rome Air Development Center, Griffiss AFB, NY, (Mar.).
- IVERSON, G.R. AND NORPOTH, H. 1976. Analysis of Variance. Sage Publications, Inc., Beverly Hills, CA.

- JEFFERY, D.R. AND LAWRENCE, M.J. 1979. "An Inter-organizational Comparison of Programming Productivity." In Proc. 4th Int. Conf. on Software Engineering, (Sept.), IEEE Computer Society Press, Los Alamitos, CA, pp. 369-377.
- JEFFERY, D.R. AND LAWRENCE, M.J. 1981. "Some Issues in the Measurement and Control of Programming Productivity." Information & Management 4, 4(Sept.), 169-176.
- JOHNSON, J.R. 1977. "A Working Measure of Productivity." Datamation 23, 2(Feb.), 106-110.
- JONES, T.C. 1978. "Measuring Programmer Quality and Productivity." IBM Sys J. 17, 1, 39-63.
- LAWRENCE, M.J. 1981. "Programming Methodology, Organizational Environment, and Programming Productivity." Journal of Systems and Software 2, 3(Sept.), 257-269.
- LYNN, C., RISLEY, J. AND WELLS, R. 1979. "Program Conversion--One Successful Paradigm." In Proc. AFIPS 1979 Nat. Computer Conf., vol. 48. AFIPS Press, Arlington, Va, pp. 139-146.
- MENDENHALL, W. 1968. Introduction to Linear Models and the Design and Analysis of Experiments. Duxbury Press, A Division of Wadsworth Publishing Co., Inc., Belmont, CA, p. 210.
- MILLS, H.D. 1980. "The Management of Software Engineering Part 1: Principles of Software Engineering." IBM System J. 19, 4, 415-420.
- MONTGOMERY, D.C. AND PECK, E.A. 1982. Introduction to Linear Regression Analysis. John Wiley & Sons, Inc., New York, NY.
- NAJBERG, A. 1981. ESD Independent Sufficiency Review of Phase IV. Air Force Electronics Division. Available from AFASPO/PGYW, Gunter AFS, AL.
- NAJBERG, A. 1983. Private Conversation. The Analytic Sciences Corp., Reading, Mass. (Oct. 12).
- OLIVER, P. 1976. Letter(Nov., 18), subject: "System Conversion." To: Deputy Chief, Program Management Office, Headquarters Air For Data Automation Agency. From: Director, Software Development Division, Automatic Data Processing Equipment Selection Office, Department of the Navy.
- OLIVER, P. 1978. "Guidelines to Software Conversion". In Proc. AFIPS 1978 Nat. Computer Conf., vol. 47. AFIPS Press, Arlington, Va, pp. 877-886.

- OLIVER, P. 1979a. "Handbook for Estimating Conversion Costs of Large Business Programs." NTIS Report AD-A065-145. (Feb.)
- OLIVER, P. 1979b. "Software Conversion and Benchmarking." Software World 10, 3, 2-11.
- PAULSEN, L. 1981. "The Implications of Program Composition and Size On Development Productivity." In Proc. 1981 Fall COMPCON, IEEE Computer Society Press, Los Alamitos, CA, pp. 149-155.
- PERLIS, A., SAYWARD, F. AND SHAW, M. 1981. Software Metrics: An Analysis and Evaluation. MIT Press, Cambridge, Mass.
- PUTNAM, L.H. 1980. Tutorial: Software Cost Estimating and Life Cycle Control. IEEE Computer Society Press, Los Alamitos, CA.
- REUTTER, J. 1981. "Maintenance is a Management Problem and a Programmer's Opportunity." In Proc. AFIPS 1981 Nat. Computer Conf., vol. 50. AFIPS Press, Arlington, Va, pp. 343-347.
- ROSCOE, J.T. 1975. Fundamental Research Statistics for the Behavioral Sciences. Holt, Rinehart and Winston, Inc., New York, NY.
- RUSHINEK, A. AND RUSHINEK, S.F. 1983. "An Evaluation of Mini/Micro Systems: An Empirical Multivariant Analysis." Data Base 14, 4(Summer), 37-47.
- SAS INSTITUTE. 1982a. SAS User's Guide: Basics. SAS Institute Inc., Cary, NC.
- SAS INSTITUTE. 1982b. SAS User's Guide: Statistics. SAS Institute Inc., Cary, NC.
- SCHNEIDER, D.B. 1978. Computer Systems Conversion - A Management Perspective. U.S. Department of Justice Report, (Oct.). NTIS No. PB-297-604.
- SCHNEIDER, G.M., SEDLMEYER, R.L. AND KEARNEY, J. 1981. "On the Complexity of Measuring Software Complexity." In Proc. AFIPS 1981 Nat. Computer Conf., vol. 50. AFIPS Press, Anlington, Va, pp. 317-322.
- SCHNEIDER, V. 1978. "Prediction of Software Effort and Project Duration - Four New Formulas." SIGPLAN NOTICES 13, 6(June), 49-59.
- SCOTT, R.F. AND SIMMONS, D.B. 1974. "Programmer Productivity and the Delphi Technique." Datamation 20, 5(May), 71-73.

- SCOTT, R.F. AND SIMMONS, D.B. 1975. "Predicting Programming Group Productivity: A Communications Model." IEEE Transactions on Software Engineering SE-1, 4(July).
- SHNEIDERMAN, B. AND THOMAS, G. 1982. "Automatic Database System Conversion: Schema Revision, Data Translation and Source-to-Source Program Transformation." In Proc. AFIPS 1982 Nat. Computer Conf. vol. 51. AFIPS Press, Arlington, VA, (June 7-10), pp. 579-587.
- SKALL, M.W. 1982. "Guide to Contracting for Software Conversion Services." National Bureau of Standards Publication 500-90, (May).
- WALSTON, C.E. AND FELIX, C.P. 1977. "A Method of Programming Measurement and Estimation." IBM Systems J. 16, 1, 54-73.
- WIENER-EHRLICH, W.K., HAMRICK, J., AND RUPOLO, V. 1981. "Applicability of the Rayleigh Model to Three Different Types of Software Projects." In Proc. IEEE 1981 Fall COMPCON. IEEE Computer Society Press, Los Alamitos, CA, (Fall), pp. 128-148.
- WOLBERG, J.R. 1981. "Comparing the Cost of Software Conversion to the Cost of Programming." SIGPLAN NOTICES 16, 4(Apr.), 104-110.
- WOLBERG, J.R. 1982. "A Costing Model for Software Conversions." Software Practice & Experience 12, 11(Nov.), 1043-1049.
- WOLBERG, J. R. 1983. Conversion of Computer Software. Prentice-Hall, Englewood Cliffs, NJ.
- WOLVERTON, R.W. 1974. "The Cost of Developing Large-Scale Software." IEEE Trans. on Computers C-23, 6(June), 615-636.
- WOODFIELD, S.N. SHEN, V.V. AND DUNSMORE, H.E. 1981. "A Study of Several Metrics For Programming Effort." Journal of Systems and Software 2, 2(June), 97-103.

GLOSSARY

ANALYSIS OF VARIANCE(ANOVA): The analysis of variance is a special case of general linear regression analysis. There are two primary uses of ANOVA: development of a regression(ANOVA) model where all the variables are of the categorical(qualitative) type and hypothesis testing during normal regression analysis. The ANOVA model typically determines the affects of the the various levels of the categorical variables and their interactions on the overall average of the dependent variable. The results of the ANOVA include the estimated coefficients(parameters or effects) of each categorical variable with associated measures of significance. The hypothesis tested by the ANOVA is that all the coefficients or parameters are 0. The F statistic is used for this test and if the probability of making a Type I error is less than the selected level of significance then the null hypothesis is rejected and significant coefficients are assumed to exist.

AVERAGE PREDICTION ERROR: The average prediction error is used when validating the predictive power of a regression model with new data. The sum of the deviations(differences between the predicted and observed values of the dependent variable) is calculated and then divided by the number of data points in the sample. It is not expected that the average prediction error be equal to zero but that it be relatively close to zero indicating approximately unbiased predictions.

AVERAGE SQUARED PREDICTION ERROR: The average squared prediction error is used when validating the predictive power of a regression model with new data. It is calculated by first determining the sum of the squared deviations(differences between predicted and observed values) and then dividing by the number of points in the sample. The result may be compared to the mean square error which can be thought of as the average variance of the residuals(deviations or errors) from the model's fit. The difference between the average squared prediction error and the mean square error should not be excessive for one to conclude that the regression model is likely to be successful as a predictor.

CATEGORICAL VARIABLE: Categorical variables are qualitative variables which fit research subjects into categories in which the notion that one category is higher than or lower than another category can not be substantiated. A typical categorical variable is that of sex--male or female.

CHI-SQUARE TEST: Chi-square tests are typically called goodness of fit tests since they are used to determine whether an observed

frequency distribution departs significantly from a hypothesized frequency distribution. Chi-square tests are also used with contingency(2 by 2) tables to determine whether two qualitative(categorical) variables are related. A measure of association(correlation) between the variables is normally computed with the Chi-square test for a contingency table. Also, a probability of a Type I error is provided to test the null hypothesis that the variables are independent or unrelated. A probability higher than one's selected level of significance indicates that the null hypothesis of independence can not be rejected.

CORRELATION: The term correlation refers to the degree of relationship or correspondence between two variables. Correlated variables are those which tend to vary together. A correlation coefficient is a measure between -1 and 1 which indicates the strength and direction of the relationship existing between the two variables. There are several different kinds of correlation coefficients but they have a common meaning. The closer to one(+1 or -1) the coefficient is, the greater the degree of the relationship or correlation. A correlation matrix provides a method for describing the correlation between pairs of several variables where the diagonal is always one.

F STATISTIC: In regression analysis, the model F statistic is determined by computing the quotient of the mean squares of regression(model) divided by the mean squares of the error(residuals). If the model provided a good fit then one would expect the F statistic to be a few times larger than the critical F value. The critical F value may be calculated by using the degrees of freedom of the mean squares, a selected level of significance, and the F distribution or table of F values. This higher F statistic allows one to reject the associated null hypothesis that all of the coefficients or parameters of the model are 0. With SAS, the F distribution probability associated with the F statistic is provided for comparison with one's level of significance such that if the probability is greater than the selected level of significance one can not reject the null hypothesis that all of the coefficients are 0. A partial F statistic is also used to test the null hypothesis that one specific coefficient or parameter is 0. The partial F statistic is provided by SAS for each of the variables in the model. If the variable being tested is x then the partial F statistic is computed by first determining the measure of the sum of squares of the model(regression) given that all the other variables(except x) are in the model; that is, the "extra sum of squares" due to x. This "partial" sum of squares is then divided by the mean squares of the error(residuals) in the model to calculate the partial F statistic.

FACTOR ANALYSIS: Factor analysis is a method used to study the interrelationship between quantitative(continuous) variables with the objective of reducing the number of variables to a smaller set that retains the original information as much as possible. The new variables(factors) are exact mathematical transformations of the original data and are constructed on the basis of the interrelations exhibited in the data. The factors are usually extracted in such a way that one factor is independent from the other.

GLM SAS OUTPUT INTERPRETATION: There are basically four parts to the GLM procedure output provided in this thesis research.

1. Regression model ANOVA results. There are three types of sum of squares provided by the basic ANOVA table: model(regression), error(residual) and corrected total(total). Each sum of squares has an associated degrees of freedom(DF). The DF for the model sum of squares are equal to the sum of the number of quantitative variables in the model plus the number of levels(categories) minus one for each qualitative variable present in the model. The DF for the total sum of squares are equal to the size of the sample(n) of data points minus one. The DF for the error sum of squares are equal to n(sample size) minus the DF for the model sum of squares minus one. The model or regression mean square is computed by dividing the model sum of squares by the associated DF. The error or residual mean square is computed by dividing the error sum of squares by the associated DF.

2. Basic model statistics. The model F statistic is calculated by determining the ratio of the model mean square and the error mean square. The probability(PR) or p value associated with the F statistic indicates the probability of obtaining this value of F or one larger by chance alone or this probability can be interpreted in relation to one's chosen level of significance, such that it indicates the probability of committing a Type I error. If the probability is higher than one's level of significance, one can not reject the null hypothesis that all the model parameters(contributions) are 0. The R-SQUARE is the coefficient of determination. C.V. is the coefficient of variation computed by dividing the square root of the error mean square(considered the average model variance of the residuals) by the mean of the dependent variable and expressing it as a percentage. This measure indicates that the residual variation is x% of the mean of the dependent variable. Finally, the overall mean of the dependent variable is provided.

3. Partial sums of squares. The TYPE III SS represent the partial sums of squares for each of the variables. They are specifically defined as the "extra sums of squares" due to the addition of the variable to the model given that all the other

variables are already in the model. The DF associated with each variable are equal to 1 for quantitative variables and the number of categories minus one for each of the qualitative variables. The F VALUE represents the partial F statistic for testing the null hypothesis that the coefficient(contribution) of the variable is 0. The probability associated with the partial F statistic has the same meaning as that of the model F statistic except the test is for a single coefficient.

4. Estimates of coefficients. The last section of the GLM output presents the model parameters. The intercept is listed first followed by the coefficients of all the variables in the model. Categorical variables are depicted with one coefficient for each level of the categories minus one. The affect of the last category of each of these variables is included in the intercept term. It should be noted that each category's coefficient is associated with a binary value, such that only the category represented by a particular subject provides an impact, that is, a value of 1 is multiplied by the coefficient.

MULTICOLLINEARITY: Multicollinearity defines the problem of linear dependencies or correlations between the independent variables in regression analysis. Since the variables are not trully independent, the method of least squares will produce poor estimates of the individual model parameters. When the variance inflation factors(see definition) exceed 5 or 10, the associated regression coefficients are poorly estimated because of multicollinearity. The simplest method for dealing with this problem is to remove one or more of the correlated independent variables and re-start the analysis.

MULTIPLE CORRELATION COEFFICIENT₂: The multiple correlation coefficient is the square root of R^2 (see definition).

PREDICTION ERROR SUM OF SQUARES(PRESS): The PRESS statistic is used as a form of data splitting, when other forms are not feasible, in model validation. To calculate PRESS, an observation i is selected and removed and the regression model is fitted to the remaining $n-1$ observations. This new model is used to predict the withheld observation y_i and the prediction error(e_i) is determined to be the difference between the actual observation and the predicted value. The same procedure is followed for each observation. Finally, the value of PRESS is calculated as the sum of all the prediction errors squared. The PRESS is then used to calculate R^{*2} for prediction which measures the model's ability to predict new observations.

R^2 : The coefficient of determination(R^2) is typically used to give the value of the model's predictive or explanatory power. Its value has a range between 0 and 1 with a value closer to 1 indicating a better model. The coefficient of determination is

calculated as the ratio of the sum of squares due to the model to the total sum of squares. Its square root is generally called the multiple correlation coefficient in multiple regression analysis.

R^2 PREDICTION: The R^2 for prediction is defined as the percent variability in the new data explained by an existing model. It is calculated as 1 minus the ratio of the error sum of squares to the total sum of squares. It is typically used to assist with the validation of an existing model with new data. This measures how well the model predicts new observations as compared to how it fits the original data.

REGRESSION ANALYSIS: Regression analysis is a statistical technique to determine the equation of the line or curve which minimizes the deviations between the observed data and the regression equation values. Regression is based on the least squares principle of minimizing the error sum of squares. The regression model that results has predictive or explanatory power which is typically measured by R^{*2} , the F statistic and the reasonableness of the parameters estimated. Given a set of variables, there is probably more than one model that fits the data well with a different group of variables in the model. If several variables are involved, the term multiple regression analysis is used.

SIGNIFICANCE LEVEL: The significance level is the degree of uncertainty about a particular statistical statement under specified conditions. Significance levels are typically signified by alpha and common values are 0.10, 0.05 and 0.01. The significance level is normally associated with the Type I error probability, such that, if the calculated probability is higher than the level of significance one can not reject the null hypothesis since it may be true.

SUMS OF SQUARES: The total sum of squares for any data may be computed as the sum of squares of the differences between each dependent variable value and the mean associated with the dependent variable. The total sum of squares has two components: the sum of squares explained by regression and the sum of squares unexplained by regression. The sum of squares explained by regression is typically called the model or regression sum of squares and is calculated as the sum of squares of the differences between the predicted y_i and the mean of $y(\text{actual})$. The sum of squares unexplained by regression is typically called the error or residual sum of squares and is calculated as the sum of squares of the differences between the predicted y_i and the actual y_i . The mean square value of each component of the total sum of squares is usually calculated by dividing each sum of square by the associated degrees of freedom. The ratio of the model or regression mean square to the error or residual mean square determines the F statistic for model hypothesis testing.

VARIANCE INFLATION FACTOR(VIF): The variance inflation factor(VIF) is typically used to detect multicollinearity. It can be shown that the variance of each estimated partial regression coefficient is "inflated" by the factor $1/(1-R_i)$, where R_i is the coefficient of determination of each of the independent variables as related to all other independent variables. The variance is larger by that factor than it would be if all independent variables were uncorrelated($R_i=0$). The SAS GLM procedure prints the reciprocal(tolerance value) of the VIF. A VIF that exceeds 5 or 10 implies multicollinearity and requires investigation.

APPENDIX A

DETAILS OF CONVERSION EFFORT/COST ESTIMATION MODELS

Federal Conversion Support Center Hybrid Model

The FCSC model was originally and formally called FCSC Hybrid Conversion Cost Model since it is a combination of several models and ideas[Federal Conversion Support Center, 1981a; 1982a]. The model was designed to cover a wide spectrum of conversion costs, from planning to system testing and documentation. The FCSC model is concerned not only with costs for staff resources but also with machine and miscellaneous resources. For those tasks that are very site dependent or unique for each conversion effort, only guidelines for costing are provided.

The FCSC model's estimate for conversion planning and analysis is a function of the size of the project and the detail of the analysis and planning required. Assuming conversion to a noncompatible target environment, the staff-days required are calculated as follows:

$$SD = 5*S + P + J$$

where,

SD = number of staff-days

S = number of systems

P = number of programs(1 SD per program)

J = number of job streams(1 SD per job stream).

For conversions to a highly compatible target environment, the staff days are reduced for each ingredient of the equation, such that

$$SD = S + P/2 + J/2.$$

For conversions to environments that have other degrees of compatibility, the number of staff-days per system may be varied between 1 and 5, and the number of staff-days per program and per job stream between 1/2 and 1.

The FCSC model presents an equation for estimating the effort involved in work package identification and preparation. The resources required, whether or not the target environment is compatible, are calculated as follows:

$$SD = 3*S + (P + F + J)/10$$

where S, P, J, and SD are defined as above and

F = number of files.

Depending on the degree of compatibility and work already accomplished the constants 3 and 10 may be changed. Note also that the estimate includes 3 staff-days for every system and 1 staff-day for every 10 system components.

The test data generation estimates, including the transfer of test files to the target machine, take into consideration the status of the documentation and the amount of code exercised by the test data. The formula is illustrated as:

$$SD = [(5*P)+(2*F)]*(TDR-TDE)*[1.0-(DOC/3)]$$

where the model estimates 5 staff-days per program(P) and 2 staff-days per file(F) and

TDR = percentage of code the test data is required to
exercise

TDE = percentage of code the test data currently exercises

DOC = percentage of adequate and up-to-date documentation.

When $TDR = TDE$ or $TDE > TDR$, it is estimated that about 1 staff-day per program will be necessary to validate the existing data and its percentage of execution ($SD = 1 * P$).

The FCSC model equation for application program and system software conversion, is based on several factors. One such basis is complexity. The model documentation provides a guideline matrix for assessing the intrinsic complexity of the software inventoried. The complexity is basically divided into 5 classes identifying programs and system software eligible for (in FCSC's terminology):

- 1) reprogramming
- 2) major program logic modification
- 3) minor program logic modification
- 4) simple syntax translation
- 5) software transference

Another basis for the equation is programmer productivity which must be quantified. In order to do this, the FCSC model uses the three development tasks of design analysis, programming and testing. A new development effort has typically been found to require 40% for design analysis, 20% for programming and 40% for testing.

From its study, the FCSC has developed assumptions as to the percentage of effort of each of these three tasks for each of the 5

complexity classes listed above. For a class 1 conversion, the total effort relative to new development is reduced to 80% and is divided as follows: 30% for design analysis, 15% for programming and 35% for testing.

To summarize all five complexity classes as listed by the FCSC one can provide a matrix of percentages such as those in Table 13

Table 13. Task Percentages for FCSC Complexity Classes

| Class | Effort Relative To New Development | Design Analysis | Programming | Testing |
|-------|---------------------------------------|--------------------|-------------|---------|
| 1 | 0.800 | 0.30 | 0.150 | 0.350 |
| 2 | 0.500 | 0.20 | 0.100 | 0.200 |
| 3 | 0.160 | 0.04 | 0.020 | 0.100 |
| 4 | 0.035 | 0.01 | 0.005 | 0.020 |
| 5 | 0.001 | 0.00 | 0.000 | 0.001 |

Another basis for program and system software effort estimation is the documentation status. In order to avoid excessive subjectivity in estimating the percentage of adequate, up-to-date documentation that exists, it is preferable to estimate the documentation status at as detailed a level as possible. However, summarizing the status of documentation on an overall system level may suffice. The FCSC model guidelines present a total of 10 documents which must be rated between 0(no documentation exists) and 10(complete set exists and is up-to-date). The maximum possible total of all would be 100%; however, the typical overall total status is well below 100%.

The last basis is the productivity rate which is highly subjective since it is usually difficult to determine what is included in the measurement; i.e., the entire project or only the conversion of the software, or manual, automatic or mixed software translation. The FCSC model uses the RADC calculated median manual productivity rate of 12.6 debugged lines of new development code per day for a development programmer.

Each FCSC complexity class may then have an average manual conversion productivity rate calculated as follows:

$$MCPR_s = \frac{BR * NDE}{[(1.0 - (DOC/2)) * DE_s] + PE_s + TE_s}$$

where,

$MCPR_s$ = average manual conversion productivity rate in number of debugged LOC manually converted per day for each complexity class

DOC = documentation status percentage(as a fraction)

BR = baseline productivity rate for new development in debugged LOC developed per day = 12.6

NDE = total effort required for new development = 100

s = complexity class: 1, 2, 3, 4, or 5

DE_s = design effort required for each class:

1 => 30; 2 => 20; 3 => 04; 4 => 01; 5 => 0

PE_s = programming effort required for each class:

1 => 15; 2 => 10; 3 => 02; 4 => .5; 5 => 0

TE_s = testing effort required for each class:

1 => 35; 2 => 20; 3 => 10; 4 => 2; 5 => .1

The FCSC model presents a table with MCPR calculated for each complexity class for various levels of documentation status.

The FCSC 1982 report also indicates that while no empirical data exists on the actual productivity rate of an automatic translator, for estimating purposes it can be assumed to be between the manual productivity of class 4 and class 5 type conversions. Therefore the FCSC uses 630 debugged LOC per day as the productivity of an automatic translator. The FCSC also assumes the following typical ranges for correct automatic translation percentages for each of the 5 classes:

| | |
|-------------------------------|----------------|
| 1 -- 0 - 25% | 2 -- 20 - 75% |
| 3 -- 65 - 90% | 4 -- 80 - 100% |
| 5 -- no automatic translation | |

The FCSC model's equation for calculating the staff-day resource requirements can now be defined. The resources required are calculated for each complexity class as follows:

$$SSD_s = \frac{LOC_s * (1 - T_s)}{MCPR_s} + \frac{LOC_s * T_s}{ACPR}$$

where

SSD_s = staff-day resources required for each complexity class (for each system, or whatever breakdown used by the estimator)

LOC_s = LOC for each conversion complexity class including comment lines for all application programs and system software

s = conversion complexity class: 1, 2, 3, 4, or 5

T_s = percentage (expressed as a fraction) of LOC capable of being correctly translated by an average automatic translator.

Typical ranges are shown above. $T = 0$ if no automatic translation is used.

MCPR = as defined earlier

ACPR = average automatic conversion productivity rate for an automatic translator = 630 LOC per day.

Finally, the total staff-day resource requirements for application program and system software conversion can be determined by summing the results of all classes as follows:

$$SD = \sum_{s=1,5} SSD_s$$

The next task for which the FCSC model provides estimates is that of data file and data base conversion. The complexity of this conversion directly impacts the total effort. Therefore, the FCSC chose to define complexity classes in this area also. Some guidelines for determining the appropriate class are provided in table form by the FCSC. The following definitions of classes apply:

- E -- A file is considered to be in class E if the source and target environments are fully compatible and conversion is not really required. In this case, a before and after file compare should be performed.
- D -- A file is considered to be of simple complexity(Class D) if the conversion is character-to-character, from source to target character set, on a one-to-one basis. Flat physically sequential files fall in this category.
- C -- Class C files are of average complexity and are involved in character-to-character, character-to-word, or word-to-word

conversions with the conversion parameters embedded in the files. Examples are compressed or variable length record files.

B -- A file is of class B(complex complexity) if the conversion required is character-to-word, word-to-word, or word-to-character with the conversion parameters external to the file. Examples are binary and floating point files.

A -- Class A(very complex conversion) files are DBMS files or data bases or can be combinations of any of the above mentioned file features.

The first step in calculating the data file and data base conversion staff-day resources required is to calculate the overall percentage of documentation(DOC). Secondly, calculate the average manual productivity rate(MCPR) for the data description or data dictionary language if they exist, for each class of 1 through 5. This is identical to the MCPR calculation given earlier for application program and system software. Thirdly, using this MCPR and the ACPR of 630 given above, calculate the staff-day resource requirements(SSD) for each software complexity class 1 through 5 using the SSD formula given earlier and let SSD equal to staff day resources for these calculations(DSD). Fourthly, classify the file and data bases by conversion complexity class of A through E as defined above and calculate the staff-day resources for each file complexity class as follows:

$$FSD_f = (F_f * FCF_f) * (1.0 - (DOC/2))$$

where,

F_f = number of files to be converted for class f

f = file complexity class: A, B, C, D, or E

FCF_f = file conversion complexity factor for each f
 class(class A=5, B=3, C=2, D=1, and E=.25)

DOC = documentation status

FSD_f = staff-day resources required for file conversion for
 each file complexity class

Finally, one must sum the FSD_f for all file complexity classes and the DSD_s for conversion of any data description or dictionary languages for the five software complexity classes as follows:

$$SD = \sum_{f=A,E} FSD_f + \sum_{s=1,5} DSD_s$$

where:

FSD_f = staff-day resources for file conversions of f classes
 as defined above

DSD_s = staff-day resources for conversion of data languages
 for each class 1 through 5 as defined for SSD_s above

SD = total effort for file and data base conversion

The next task receiving attention by the FCSC model is that of operation control language(OCLE) conversion. The methodology used for estimating OCL conversion is identical to that used for application program and system software conversion. The lines of OCL code are used as input for estimation. Based on the assumption that each job stream consists of lines of OCL, the OCL can be estimated by assuming an average number of lines of OCL for each job stream. After the lines of OCL are calculated, one calculates the overall percentage of

documentation(DOC) and the MCPR for each class. The staff-day resources(SSD) for each class are calculated and then summed as before to calculate the staff-day resources(SD) for the OCL conversion for all classes. The formulas are the same as those for application program and system software conversion.

The next task estimated is that of system testing which is defined as the full application system testing using test data. The duration of the system testing is compounded by a rerun factor due to the fact that the testing may have to be restarted many times. The staff-days(SD) required, assuming a non-compatible target environment, are estimated to average about 1 SD per 4 job streams(J), 1 SD per 2 programs(P), 1 SD per 2 systems(S) and 1 SD per 80 system components(P+F+J, where F=files). This is multiplied by a rerun factor(RE) which indicates the number of probable reruns necessary to achieve a successful test. Typically, conversions require 5 to 10 reruns. The staff-days(SD) are thus calculated as follows:

$$SD = [J/4 + P/2 + S/2 + ((P+F+J)/80)] * [1 + (RE/10)].$$

In the case of a more compatible target environment less testing is anticipated before reaching acceptable output. In this case, testing is to average 1 SD for every 10 systems(S) and files(F) and 1 SD for every 80 system components. This is multiplied by a rerun factor(RE) that is lower than for a non-compatible case. Thus for a more compatible environment one uses:

$$SD = [((S+F)/10) + ((P+F+J)/80)] * [1 + (RE/10)].$$

The next task detailed by the FCSC model is that of redocumentation. This redocumentation refers not to specific unit tasks but rather to overall system and project level redocumentation effort required. It is assumed that both technical and clerical staff are required here. Assuming conversion to a non-compatible target environment, the technical staff-day (TSD) resource required for the technical portion of the redocumentation requires approximately 1 staff-day for every 4 programs(P) and 1 staff-day for every system(S). A percentage factor(RCOR), typically 10%(expressed as a fraction), is added to the formula. RCOR represents the coordination effort among the technical, clerical and the entire project staff. Documentation percentage status(expressed as a fraction) is also included. Thus,

$$TSD = (P/4+S)*RCOR*DOC.$$

The clerical staff resource requirements(CSD) require approximately twice as long as those for the technical staff as follows:

$$CSD = (P/2+2*S)*RCOR*DOC.$$

Thus the total staff days(SD) for redocumentation are calculated as follows:

$$SD = TSD + CSD.$$

This total could be adjusted if a high degree of compatibility were present.

The next FCSC model task is acceptance testing which involves the converted programs, revised documentation and procedures and converted live data. The objective of this test is to achieve an acceptable comparison of outputs against the source system results.

Typically, acceptance testing requires a basic level of staffing for the entire duration of the test and a high level during the beginning of the test cycle. Therefore, an exponential function is required to express this pattern.

Assuming conversion to a noncompatible target environment, the staff-days(SD) resources required for the basic or constant level of staffing is usually about 1 staff-day for every 20 systems(S) for the duration(DUR) of the test. In acceptance testing, only the programs(P) and files(F) grouped together need to be tested. The resources required are about 1 SD for every 5 programs and files. This is multiplied by a negative exponential function to temper the effect of a long duration. Thus,

$$SD = [DUR*(S/20)] + [(P+F)/5] * (1 - e^{-(DUR/20)}).$$

The next three tasks of the FCSC model are site preparation, system transition(complete parallel, immediate cutover, or phased parallel) and training. No estimating equations are provided since there is a great deal of variability in these three tasks for different organizations, environments and conversion efforts.

The next task addressed by the FCSC model is that of conversion management and administrative overhead and/or contract administration and support. The management to technician ratio(typically 1:10) must be determined by the organization to perform the conversion. This ratio is only concerned with the organization's own labor force. The contractor's management is absorbed in contractor rates, if any activity is contracted out.

The management to technician ratio(MTR), expressed as a fraction, is applied to the total in-house staff resources(TINSD) for the following tasks only:

- * conversion planning and analysis
- * conversion work package identification and preparation
- * test data generation and validation
- * application program and system software conversion
- * data file and data base conversion
- * operation control language conversion
- * redocumentation
- * system testing
- * acceptance testing.

For those tasks that are contracted, an additional 10% is added for contract administration and support. This 10% is applied to the total contractor staff-day resources(TCONSD) estimated. Therefore, the total resources for conversion management and administrative overhead are computed as follows:

$$SD = (TINSD * MTR) + (TCONSD * .10).$$

The FCSC model briefly discusses conversion aids since they can significantly reduce the time and cost of a conversion project. There are no typical costs for conversion tools. Each case must be separately addressed so no formulas can be derived for this task.

Once the staff-day resource requirements for the baseline conversion tasks are completed, their costing may commence. The FCSC 1982 report contains salary figures for various grades of federal

civilian employees. These could be used to calculate the average in-house personnel rate(IN\$). An average contractor rate(CON\$) for a typical conversion effort in 1982 is about \$60,000 per staff year or \$280 per staff-day. The FCSC uses 213.2 staff-days per year and 17.77 staff-days per month. The staff resources costing(SCOST) is then based on the total staff-days(SD) and the percentage of work to be done by in-house resources(IN%):

$$SCOST = (SD * IH\% * IH\$) + (SD * (1 - IH\%) * CON\$).$$

For machine resource estimating and costing, the FCSC provides a table which illustrates the baseline conversion tasks and the expected number of machine hours a conversion programmer is assumed to use per staff-month of effort. The machine resources in machine-hours(MH) are calculated by dividing the total SD for each task by 17.77 and multiplying the result by the number of machine-hours assumed to be used per staff-month(MHR). This is illustrated as :

$$MH = (SD / 17.77) * MHR.$$

The machine resources in machine cost(MCOST) are calculated by multiplying MH by the percentage of use for the source machine(SM%) and the target machine(TM%) and multiplying the results by the average hour rate for each machine(SM\$ and TM\$). Thus,

$$MCOST = (MH * SM\% * SM\$) + (MH * TM\% * TM\$).$$

The last item included in the FCSC model is miscellaneous resource estimating and costing. No estimating equations can be given for this area for obvious reasons. Some items to be considered under this task are suggested in the FCSC document.

Hahn and Stone Model

The earliest attempt to define a parametric conversion cost estimation technique appears to be that Hahn and Stone of MITRE Corporation[Hahn and Stone, 1970]. The cost estimation model includes three cost categories: cost of transferring programs(C_P), cost of transferring data(C_D) and other costs(C_O) such as documenting the programs. The model is represented by an equation for total cost(C_T):

$$C_T = C_P + C_D + C_O$$

The cost of transferring programs is made up of two major components: the cost of manually transferring(C_M) and the cost of automatically transferring(C_A) all or part of a program. This cost is expressed as :

$$C_P = C_A + C_M$$

The cost of automatic transfer(C_A) can not be further defined since it will vary with the type of transfer technique(redesign, reprogramming, or recoding) and the differences between the source and target environments.

The MITRE report defines the cost of manually transferring programs as the largest single cost of the transfer process. This cost(C_M) uses three items in its calculation: (1) number of instructions(lines) which must be manually transferred(I); (2) the rate at which this can be done(lines/man-day)(R_T); and (3) cost of manpower per man-day(C_{MD}):

$$C_M = (I/R_T) * C_{MD}$$

In order to determine an equation for calculating the rate of transfer(R_T), Hahn and Stone of MITRE performed an extensive review of the literature on estimation related to new developments, analyzed program development data from several sources including the federal government and MITRE. It was found that the number of statements which can be manually transferred from one computer to another in one man-day is a function of several variables as described below.

The basis for analysis was determined to be the new development production rates. Some "rules of thumb" from the literature indicated rates of one instruction/hour(8 instructions per man-day) and 200 instructions/man-month(10 instructions per man-day). The development data analyzed resulted in estimates of production rates(instructions/man-day) for FORTRAN, COBOL and JOVIAL, whose respective mean rates were 4.5, 5.8 and 5.7. Since conversion of an existing program does not include all the functions normally associated with development projects, the above production rates had to be modified to establish a baseline conversion production rate(R_{BC}) and a program testing production rate(R_{BT}).

By studying the division of effort for the tasks required for new developments and the data of some conversion projects, a division of effort was calculated for redesign, reprogramming, recoding and conversion program testing. The conversion production mean rates were then calculated and are shown in Table 14.

To compensate for critical aspects of the conversion effort, a number of degradation factors were developed which add to the total

Table 14. Hahn and Stone Conversion Production Mean Rates

| | R_{BC} Redesign | R_{BC} Reprogram | R_{BC} Recode | R_{BT} Test |
|----------|----------------------|-----------------------|--------------------|------------------|
| FORTTRAN | 8.2 | 11.3 | 22.5 | 14.1 |
| COBOL | 10.5 | 14.5 | 29.0 | 18.3 |
| JOVIAL | 12.5 | 17.3 | 34.5 | 21.7 |

effort required to transfer programs. Complete, accurate and up-to-date documentation is necessary for efficient transfer of programs. A suggested list of documentation that should be available is provided in the report. The documentation factor(D_{F1}) requires an increase in manpower for various documentation status categories. These are shown in Table 15 below.

Table 15. Hahn and Stone Documentation Status Categories

| Category | % Increase in Effort(D_{F1}) |
|-----------|----------------------------------|
| Excellent | 0% |
| Good | 25% |
| Average | 50% |
| Poor | 75% |

A program instability factor(D_{F2}) is used to account for the increase in effort necessary if modifications of the program will take place during the transfer. The extent of past modifications of the program is used as a guide to determine a modification level for a D_{F2} rating. The levels suggested are shown in Table 16

Table 16. Hahn and Stone Modification Level Ratings

| Modifications | % Increase in Effort(D_{F2}) |
|---------------|----------------------------------|
| Nil | 0% |
| Trivial | 5% |
| Some | 10% |
| Extensive | 15% |

A third degradation factor used in the MITRE model is that of system integration(D_{F3}). Program complexity has some impact on the amount of resources required to transfer programs from one environment to another. Based on some data analysis, Hahn and Stone propose a degradation factor of $.016*N$ where N is the number of subprograms in the programs being converted. This factor becomes a percentage that is used to increase the amount of testing required.

Now the total rate of transfer may be defined as

$R_T = I/MD_T$ where MD_T is the total number of man-days required. With the factors given above then:

$$MD_T = I/R_{BC} + (D_{F1} * I/R_{BC}) + (D_{F2} * I/R_{BC}) + I/R_{BT} + (D_{F3} * I/R_{BT}) \text{ where,}$$

I/R_{BC} = man-days for baseline conversion

$D_{F1} * I/R_{BC}$ = mandays for documentation degradation factor

$D_{F2} * I/R_{BC}$ = mandays for program instability factor

I/R_{BT} = mandays for baseline test

$D_{F3} * I/R_{BT}$ = mandays for system integration factor

Substituting MD_T into the R_T equation results in:

$$R_T = \frac{R_{BC} * R_{BT}}{R_{BC}(1+D_{F1}+D_{F2}) + R_{BT}(1+D_{F3})}$$

The data transfer costs are typically small in comparison to program transfer costs. No specific method of computation is provided by Hahn and Stone. The other costs involved in a transfer are quite numerous and varied so the model only mentions some functions that must be considered for a total costing calculation. Some of these are training, facilities, planning and management.

The Hahn and Stone or MITRE model can then be summarized as follows:

$$C_T = \sum I * \frac{R_{BC}(1+D_{F3}) + R_{BT}(1+D_{F1}+D_{F2})}{R_{BC} * R_{BT}} * C_{MD} + C_D + C_O$$

The summation symbol indicates the sum over all the programs in the inventory and all the symbols retain their meaning as defined above. It should be reemphasized that I represents only the LOC to be manually converted.

Grim, Epler and Andrus Estimation Method

Grim, Epler and Andrus reviewed the Air Force procedures for conversion cost estimation and found that there was no comprehensive method for estimating such costs. The results of their review and study was a document suggesting guidelines for the costing of conversion efforts[Grim, Epler and Andrus, 1978].

The first element examined was the cost of converting applications programs which can be divided into analysis, programming, manpower and machine costs. To calculate analysis

costs, one must first determine if there are patches. If so, one must allow one man-day per 10 patch lines of code. Another analysis cost element is that of sorts. If the target environment does not produce sorted output identical to that of the source then allow a maximum of two man-days of analysis per sort per program.

Site unique utilities must be converted if not replaced by vendor software and if to be converted allow analysis time of between two to five man-days for each utility. If the target computer does not have equal or greater precision than the source, two man-days are allowed for each program that performs arithmetic calculations.

After the analysis cost is determined, the programming cost in man-days(M) must be calculated using the equation:

$$M = \frac{2 * N_s * (1 - \%T)}{R_s * (1 + D)}$$

where,

N_s = number of source lines to be converted

$\%T$ = percentage(expressed as a fraction) of LOC translated by a automatic translator

R_s = number of LOC converted per day by the average programmer.

This factor may be locally derived or one of the factors provided in the study may be used. Examples of factors provided are : COBOL to COBOL translations approximate 30 lines of old system per day while COBOL to FORTRAN approximates 10.5 lines per day.

D = percentage that the old system is documented expressed as a decimal. A value of 1.00 represents complete and up-to-date

documentation. No documentation or comments in a program means a value of 0.0 is appropriate.

After M above is computed, it may be increased up to 20% if the programs in question are thought to be more complex than the average. To determine the dollar programming manpower cost(C_p), the organization's dollar cost for one programmer man-day(P_M) is multiplied by M:

$$C_p = M * P_M$$

To determine machine costs(C_M), several programs of average size are compiled and test run and an average cost per run(R) is calculated. In a batch environment the average programmer makes two runs per day so

$$C_M = 2 * M * R.$$

In an online programming system, 3.5 runs per day is appropriate so:

$$C_M = 3.5 * M * R.$$

After examining application program conversion costs, data conversion is analyzed. The basic cost factor in data conversion is the cost of reading or writing a single average size physical record of data which is the cost of a single computer input/output unit. This cost may be determined from the old system. This old system I/O unit cost(I_o) should be multiplied by the average number of physical records per file(F_r) and then multiplied by the number of files(N_f). To this amount one must add a similar computation for loading the file to the new system using the new system I/O unit cost(I_n). Add

to this new amount the cost of two programmer man-days($2 * P_M$) per file.

The resulting data conversion cost(C_d) is then:

$$C_d = (I_o * F_r * N_f) + (I_n * F_r * N_f) + (2 * P_M * N_f).$$

This is appropriate for sequential files. If the current files have a more complex structure than strictly sequential, the I_n and I_o costs should be multiplied by some complexity factor. This factor may be determined by comparing the cost of a sequential file dump and a complex file dump of similarly sized files.

If the target environment is an upgrade in the same product line and if a direct conversion utility is available, a new I/O unit cost(I_n) should be calculated and then:

$$C_d = (I_n * F_r * N_f) + (2 * P_M * N_f)$$

If test data needs to be generated, then analysis time of two to five man-days per input file must be considered. The actual data production should be estimated to occur at 50 to 100 records per day.

The next cost category is that of operating procedures conversion. Associated costs are difficult to quantify; however, some guidelines are provided. If the JCL on the source and target machines are similar and the differences are documented then one should allow one programmer man-day for converting the JCL of eight programs. If M is the number of programmer man-days required and N_p equals the number of programs then $M = 1/8 * N_p$. If the JCL's are not similar then $M = 1/3 * N_p$; that is, one programmer can convert the JCL for three programs in one day. The dollar cost then is

$$C_o = M * P_M$$

Other cost categories which are mentioned but for which no equations are provided are: support software conversion, facilities, training, acquisition activities, and management and administration.

AFASPO Phase IV Estimation Method

The AFASPO suggested a modified version of the Hahn and Stone Model for use in calculating an estimate of man-years required to translate those portions of programs/systems that were not successfully translated by the automatic translator[Air Force Automated Systems Project Office, 1982a].

The documentation categories of excellent(0%), good(25%), average(50%) and poor(75%) were used to adjust upwards the manpower efforts to manually transition software. Since almost all of the Phase IV effort is a COBOL to COBOL conversion, only the COBOL productivity rates were provided: Recoding(29 LOC/man-day), reprogramming(14.5 LOC/man-day) and testing(18.3/man-day). The redesign rate was not provided since redesign is not allowed for COBOL programs in Phase IV.

This first step is to calculate the total LOC to be manually converted:

$$x = a - a*b$$

where:

a = total LOC for a given software

b = translator effectiveness(90% in Phase IV)

x = LOC to be manually converted(I of MITRE model)

Then to calculate the total man-years to manually convert and test a system the AFASPO suggests using x above and:

$$y = \frac{x/d + (x*c)/d + x/e}{20 * 12} * 1.1$$

where:

y = total man-years to manually convert the software

c = factor for documentation condition (D_{F1} of MITRE)

d = productivity in LOC/man-day for manual conversion (R_{BC})

e = productivity in LOC/man-day for testing (R_{BT})

$20*12$ = productive man-days per year

1.1 = factor to adjust for cost of documentation

Since the AFASPO's primary concern was to provide software cost estimation guidelines, no other conversion costs were discussed. The few cases of FORTRAN and Burroughs Assembly language systems which are to be converted to COBOL are treated as new developments. The AFASPO summarizes Wolverton's and Aron's methods and suggests that the average of both methods be used as an estimate for these few cases [Aron, 1969; Wolverton, 1974].

Wolberg's Model.

Wolberg analyzed a group of Rand Information Systems (RIS) conversion projects and developed a cost estimation model which is included in his recent book [Wolberg, 1983]. RIS has been specializing in conversions since 1968 and has developed a variety of conversion tools and aids which would cause the RIS effort and

duration data to be lower than if the same conversions were done by relatively inexperienced personnel.

The RIS data consisted of nine completed conversion projects ranging in project efforts from 59.1 to 343.8 person-months (assuming 173.2 person-hours per person-month) and ranging in project duration from 7 to 32 months. Though various languages are represented, the emphasis is on COBOL which is not surprising since, as Wolberg states, more than 50% of the usage on mainframe computers is for COBOL programs. A second set of RIS data includes duration information for 31 projects but no effort information.

There is considerable scatter in the data, so a straight line was chosen as the most reasonable model for representing the functional relationships. A least squares solution for effort in person-months(E) is

$$E = 7.14 * L^{0.47}$$

where L = thousands of LOC to be converted.

The least squares solution for duration in months(D) is

$$D = 4.1 * L^{0.22}$$

where L is the same as above.

Note that L, in the models, represents the total lines of code to be converted. No mention is made of manual conversions. Since these are RIS projects, one can assume that automatic translators and other aids were used. However, the models call for the total LOC and not only that portion of the total that is not converted successfully by the automatic translator. It should also be mentioned that these

models are for the conversion category of recoding and that the estimates derived include tasks associated with the entire conversion project from planning to implementation.

Wolberg considered reprogramming and redesign as replacement alternatives rather than conversion options. By his definition, conversion implied that some degree of automation is possible thus recoding is conversion. However, since redesign and reprogramming are primarily manual operations, Wolberg classified them as replacement alternatives. These definitions or re-definitions impact the development of estimation equations only in the sense that Wolberg assumed that the redesign of a system requires an effort comparable to the development of a new system of the same size. In addition, Wolberg assumed that reprogramming requires an effort equal to one-half the effort required for redesign.

Wolberg used the model for new development derived by Walston and Felix to estimate the effort for a redesign project [Walston and Felix, 1977]:

$$E_{RD} = 5.2 * L^{0.91}$$

Wolberg stated that the distinction between redesign and reprogramming was not clear; however, it was clear that reprogramming implied a smaller effort. He used his 50% assumption and applied this to the redesign model to yield a reprogramming model:

$$E_{RP} = 2.6 * L^{0.91}$$

The effort E_{RD} signifies the person-months required to redesign a system of L thousands of LOC and E_{RP} signifies the person-months to reprogram a system of L thousands of LOC.

Wolberg also made an assumption that the duration of a redesign is the same as new system development and therefore used the Walston and Felix duration model for redesign estimation:

$$D_{RD} = 4.1 * L^{0.36}$$

He also made an assumption that the duration for reprogramming is 80% of the new system development or 80% of the duration for redesign. Thus, the duration of reprogramming was defined as:

$$D_{RP} = 3.3 * L^{0.36}$$

Regarding the model for recoding presented initially, Wolberg stated that since the smallest system in the RIS data base was 32,000 lines, the model might not be applicable to smaller conversion or recoding projects.

APPENDIX B

DATA ENCODING AND PRELIMINARY ANALYSIS

Introduction

The general approach to the productivity analysis was a two step process. First, the programmer resume and the program information data were studied separately. This step is detailed in this Appendix. Secondly, the separate data files were integrated, as appropriate, for further detailed analysis. This expanded work is discussed in Chapter 5. The preparatory work and preliminary analysis presented in this Appendix includes encoding of data, building of data records/files, and basic analysis of programmer resume and program information data.

Data Encoding and Preparation

The goal of the encoding was to capture as much of the original raw data as possible from the programmer resume form(Figure 1, Chapter4) while keeping in mind the purpose of the data. To determine the scheme for data encoding, the responses on 100 programmer resume forms were reviewed and analyzed. Lists of responses for all the questions were compiled and appropriate summary or explanatory codes were devised to capture the raw information. The following comments address the encoding of each specific data element.

The programmer code was originally intended to be three characters with the first character representing the Major Air Command(MAJCOM) or Separate Operating Agency(SOA) in the Phase IV conversion. This one character code (hereinafter referred to as MAJCOM), letter A through Z or number 5 to 9, was originally assigned by the AFASPO to each organization. The second and third characters of the programmer code were to represent the programmer(PGMID); however, since one of the centers encoded their programmers with four characters a total of five is required for the programmer code(1 for the MAJCOM and 4 for PGMID).

In the education section, the question of college graduate and degree produced a variety of responses. Table 17 presents the codes and corresponding graduate/degree categories which were designed to capture every conceivable input. If more than one degree was indicated, only the highest degree was encoded. Thus the graduate/degree category required one numeric character.

Table 17. College Education Categories.

| Education/Degree | Code |
|------------------|------|
| PhD | 8 |
| Post Master's | 7 |
| Master's | 6 |
| Post Bachelor's | 5 |
| Bachelor's | 4 |
| Post Associate's | 3 |
| Associate's | 2 |
| Some College | 1 |
| None | 0 |

AD-A145 757

A METHODOLOGY FOR THE ANALYSIS OF PROGRAMMER
PRODUCTIVITY AND EFFORT ESTI..(U) AIR FORCE INST OF
TECH WRIGHT-PATTERSON AFB OH J D FERNANDEZ MAY 84

3/3

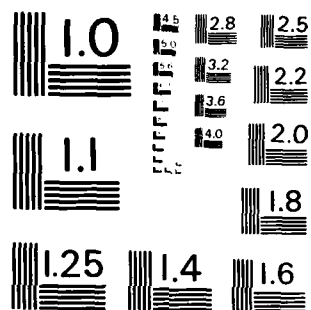
UNCLASSIFIED

AFIT/C1/NR-84-440

F/G 9/2

NL

| | | | | | | | | | | | | | |
|--|--|--|-------|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | | | END | | | | | | | | | | |
| | | | DATA | | | | | | | | | | |
| | | | FILED | | | | | | | | | | |
| | | | 10 84 | | | | | | | | | | |
| | | | DTIC | | | | | | | | | | |



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

The college major and minor questions also produced various responses. Table 18 shows the major/minor categories that were thought to capture the significant fields of specialization. Both majors and minors used the same scheme and were thus allocated one numeric character each.

Table 18. Academic Majors and Minors.

| Major/Minor | Code |
|--------------------------------|------|
| Computer Science | 6 |
| Business DP/MIS | 5 |
| Math & Engineering | 4 |
| Physical & Biological Sciences | 3 |
| Management or Other Business | 2 |
| Social Science & Other | 1 |
| None | 0 |

The question on formal instruction produced the greatest and most varied spectrum of responses. Since COBOL, COBOL-74, AFOLDS and Sperry-Univac training were known to be of interest, they were specifically broken out. A caution in this area is that an individual which showed no or little formal training may have had excellent academic courses paralleling the formal training. Table 19 shows the binary categorical variables chosen to represent an individual's formal training. A code of 1 means that the individual had formal training in that category.

The programmer resume form questions A, B, and C, under "Background", requested quantitative responses of number of years, thus encoding was unnecessary. Two numeric characters were allowed

Table 19. Formal Training Categories.

| Areas of Formal Instruction | Binary Variable |
|---|-----------------|
| COBOL-74 | FTNG6 |
| COBOL(all except COBOL-74) | FTNG5 |
| Air Force AFOLDS | FTNG4 |
| Other Languages, DBMS and General Programmer Training | FTNG3 |
| Sperry-Univac Training | FTGN2 |
| Software Engineering and other HW/SW Air Force or Vendor Training | FTNG1 |

for each response. Question D attempted to qualify the experience presented in the preceeding questions. The response to this question basically indicates the respondent's programmer type. Table 20 shows the categories of experience or programmer types possible.

Table 20. Programmer Experience Categories.

| Programmer Experience Title | Code |
|--|------|
| No Experience | 0 |
| Maintenance (If Development Exp. % is 0-33) | 1 |
| Both Maintenance and Development (If Development Exp. % is 34-67) | 2 |
| Development (If Development Exp. % is >67) | 3 |

The wording used in question E caused some confusion. The first part of the question states: "If the majority of programs you shall be transitioning are not COBOL, then what type of system are they ____?" Some responded "COBOL", others "N/A" and a few others another

language. The problem here was that an "N/A" response could mean "COBOL" while in other cases it might mean the programmer does not know. Table 21 indicates the program/system language type encoding used. The four languages shown in the table are the only ones possible in Phase IV. The second part of the question was allotted two numeric characters to represent experience.

Table 21. Programming Language of Programs to Convert.

| Source Language | Code |
|-----------------|------|
| COBOL(or N/A) | 4 |
| AFOLDS | 3 |
| FORTRAN | 2 |
| Assembler | 1 |

Questions F and H were combined for the encoding since the responses in question H overlapped and duplicated some of those in question F. Table 22 shows the categories of transition experience that captured the responses provided for these two questions. The early Phase IV experience category captures the responses that indicate participation in an early Sperry-Univac operational test evaluation as well as a programmer's opportunity to work at a center, different from his own, where a Univac system was installed for initial testing and/or conversion.

The question(G) on JCL experience was allowed two numeric characters and no encoding. In some cases the JCL experience could have been from an academic institution since the experience quantity was greater than of questions A through C which reflect years of field experience.

Table 22. Conversion Experience Categories.

| Previous Conversion Experience | Code |
|--|------|
| COBOL-68 to COBOL-74 Experience (other than Code 4 below) | 5 |
| Early Phase IV Experience | 4 |
| Other Conversion Experience-Considerable | 3 |
| Other Conversion Experience--Some/Little | 2 |
| System/Equipment Knowledge & Experience | 1 |
| None | 0 |

Except for the language used, none of the responses on the program information form required encoding. The language code was taken from Table 21. The Batch, Online and Difficulty categories were all designated as binary variables. Both Batch and Online were included because there are some systems that utilize both.

Data Entry/Data Records

The programmer resume data were first encoded and then keyed into an IBM PC which was used for the basic data entry and initial data editing process. Programmer resume data records (Table 23) were built on a 320K diskette and the IBM PC line editor was used for data corrections [IBM, 1981]. Printouts of the files were produced on an EPSON MX-80FT attached to the PC. These raw listings were used for desk checking the data. Once the files were cleaned, they were transmitted to a main frame (Amdahl) where the Wylbur text editor was used to collect the records.

Table 23. Programmer Resume Data Record.

| Field | Bytes | Variable |
|--------------------------------|-------|----------|
| Programmer Code | | |
| MAJCOM/SOA | 1-1 | MAJCOM |
| Programmer ID | 2-5 | PGMID |
| College Education | 6-6 | DEGREE |
| Major | 7-7 | MAJOR |
| Minor | 8-8 | MINOR |
| Formal Training | | |
| COBOL-74 | 9-9 | FTNG6 |
| Other COBOL | 10-10 | FTNG5 |
| AFOLDS | 11-11 | FTNG4 |
| Other Programmer Training | 12-12 | FTNG3 |
| Sperry-Univac Training | 13-13 | FTNG2 |
| Other Related Training | 14-14 | FTNG1 |
| Total Years in Computer Field | 15-16 | TOTEXP |
| Total Programming Years | 17-18 | PGMEXP |
| COBOL-68 Years of Experience | 19-20 | C68EXP |
| COBOL-74 Years of Experience | 21-22 | C74EXP |
| Programmer Experience Type | 23-23 | PTYPE |
| Source Language of Programs | 24-24 | CONLAN |
| Experience with these Programs | 25-26 | SYSEXP |
| Conversion Experience | 27-27 | CONEXP |
| Years of JCL Experience | 28-29 | JCLEXP |

The program information form content was modified slightly during data entry. AFASPO personnel commented on an early proposed data record format stating that the DSD and System Title were superfluous and unnecessary. The system code and program title(or program code) are all that is required to uniquely identify a record. In many cases there is even overlap here since the system code is part of many program codes. Table 24 shows the program information basic record. As was done with the programmer resume record, the MAJCOM code is extracted from the programmer codes provided in the activity matrix. The MAJCOM code becomes the leading character of

Table 24. Program Information Data Record.

| Field | Bytes | Variable |
|--|-------|-----------|
| MAJCOM/SOA | 1-1 | MAJCOM |
| System Code | 2-3 | SYSCD |
| Program Title | 4-9 | PROID |
| Date Started | 10-15 | STDATE |
| Start Lines of Code | 16-20 | STLOC |
| Data Completed | 21-26 | FIDATE |
| Finish Lines of Code | 27-31 | FILOC |
| Type Information | | |
| Source Language of Program | 32-32 | LANG |
| Batch Type | 33-33 | BATCH |
| Online Type | 34-34 | ONLINE |
| Difficulty | | |
| Sort | 35-35 | SORT |
| Zip | 36-36 | ZIP |
| Switches | 37-37 | SWITS |
| Comp Data | 38-38 | CDATA |
| Call | 39-39 | KALL |
| Reel#'s | 40-40 | REELNR |
| Random I/O | 41-41 | RANDIO |
| Copy(libraries) | 42-42 | COPY |
| Interrogate | 43-43 | INTER |
| Number of Programmers Assigned | 44-44 | NRPGM |
| Activity Matrix | | |
| Programmer One ID | 45-48 | PA |
| Documentation | 49-51 | HRA1 |
| Data File Transfer | 52-54 | HRA2 |
| ADS Translation | 55-57 | HRA3 |
| Create Control Language | 58-60 | HRA4 |
| Test/Debug | 61-63 | HRA5 |
| Miscellaneous | 64-66 | HRA6 |
| Knowledge Code(Knowledge of | 67-67 | KCA |
| Program: 0=not at all... | | |
| 6=wrote program) | | |
| Programmer Two ID & ETC. | 68-90 | PB |
| | | HRB1-HRB6 |
| | | KCB |
| Programmer ETC. | | |
| As many programmer sets as specified in NRPGM. | | |

the record key and the programmer codes in the activity matrix are reduced by one character. Notice that a variable is added before the activity information to indicate the number of programmers that worked on the program. The variable of number of programmers is used in building the records of the program information file. The size of the records is variable depending on the number of programmers included. Since the length of the records may be greater than Wylbur's 133 character maximum, TSO was used to receive the records transmitted from the IBM PC.

Basic Summary of the Data Base

Because of its power, flexibility and ease of use, the Statistical Analysis System(SAS), was used for the analysis of the data[SAS Institute, 1982a; 1982b]. Included in this section is the first step of the general approach of the analysis methodology. The programmer resume and program information files were viewed and analyzed separately.

Programmer Resume Data Analysis

The initial data submitted by the conversion centers was that of programmer profiles or resumes. Over 320 programmers are involved in the conversion effort. The exact number has varied slightly due to personnel turnovers and new hires. The following information describes the programmer resume file.

The simplest description of the programmer file is that of programmer type or programmer experience type. Table 25 presents the programmers as they see themselves and their experience.

Table 25. Types of Phase IV Programmers.

| Experience Type | Frequency | Percent |
|--------------------|-----------|---------|
| No Experience | 31 | 9.5 |
| Maintenance | 78 | 24.0 |
| Both Maint. & Dev. | 132 | 40.6 |
| Development | 84 | 25.8 |
| | 325 | |

A description of the programmer's academic background is the first part of the programmer resume form. Using the encoding presented earlier, the degree information sorts into the categories as shown in Table 26. Notice that there are no programmers classified as having PhD or Post Master's education.

Table 26. College Education of Phase IV Programmers.

| Education | Frequency | Percent |
|------------------|-----------|---------|
| None | 141 | 43.4 |
| Some College | 50 | 15.4 |
| Associate's | 34 | 10.5 |
| Post Associate's | 9 | 2.8 |
| Bachelor's | 72 | 22.2 |
| Post Bachelor's | 7 | 2.2 |
| Master's | 12 | 3.7 |
| | 325 | |

Computer science was the most frequently cited major field of academic specialization. Table 27 shows the majors and minors as summarized in the categories chosen during encoding. Since the academic major is the best indication of a person's background, only the major was used in the analysis.

Table 27. Summary of Majors of Phase IV Programmers.

| <u>Majors/Minors</u> | <u>Frequency</u> | <u>Percent</u> |
|-----------------------------|------------------|----------------|
| Computer Science | 49 | 15.1 |
| Business DP/MIS | 27 | 8.3 |
| Math/Engineering | 17 | 5.2 |
| Physical Biological Science | 5 | 1.5 |
| Management & Other Business | 24 | 7.4 |
| Social Science and Other | 28 | 8.6 |
| None | 175 | 53.8 |
| | <u>325</u> | |

The formal training or instruction section of the programmer resume form permitted the greatest latitude of responses. The encoding of the responses produced the classification of training as shown in Table 28.

Table 28. Formal Training Profile of Phase IV Programmers.

| <u>Formal Instruction Area</u> | <u>Frequency</u> |
|--------------------------------|------------------|
| COBOL-74 | 16 |
| All Other COBOL Training | 162 |
| AFOLDS | 49 |
| Other Programmer Training | 239 |
| Sperry-Univac Training | 283 |
| Other AF/Vendor ADP Training | 167 |

Of special interest in this endeavor is the conversion experience of programmers. Table 29 presents the conversion or transition experience as given by the programmer in response to questions F and H of the programmer resume. Notice that some considered their knowledge of the system as conversion experience.

Table 29. Conversion Experience of Phase IV Programmers.

| Conversion Experience Type | Frequency | Percent |
|--|-----------|---------|
| COBOL-68 to COBOL-74 Experience | 5 | 1.5 |
| Early Phase IV Experience | 48 | 14.8 |
| Other Conversion Experience-Considerable | 13 | 4.0 |
| Other Conversion Experience-Some/Little | 55 | 16.9 |
| System/Equipment Knowledge & Experience | 8 | 2.5 |
| No Experience | 196 | 60.3 |
| | 325 | |

For nominal or categorical type variables such as those presented above, the most commonly used analysis technique is that of the Chi-Square test of independence which is typically accomplished by means of two-way tables[Roscoe, 1975]. This test was accomplished on all variable pairs of interest. As mentioned earlier, the academic minor and responses to question E (system or conversion language and associated experience) were excluded from this analysis. The results of the Chi-Square tests revealed that the categorical variables were defined with too many levels. Almost all of the tests were invalid because the matrix or two-way tables were very sparse or over 20% of the cell counts were less than five. After reviewing the literature and studying the percentages of various categorical

levels, some recombination of categories or levels was thought to be appropriate. Table 30 shows how the college education(DEGREE), major(MAJOR) and conversion experience(CONEXP) categories were regrouped for analysis purposes. These new categories also appeared to be more meaningful.

The newly regrouped categorical variables were subjected to the Chi Square test. Table 31 summarizes the tests of the null hypothesis (H_0) that the variable pairs are independent. The level of significance or probability of Type I error (rejection of a true hypothesis) chosen for these tests was .10. Notice that the correct terminology when probability of a Type I error is greater than .10 is that one "can not reject H_0 ". This is true because the Chi-Square test does not prove independence but only allows one to assume that independence is probable. The SAS tests also provide measures of association between the variables.

The quantitative variables of years of experience were analyzed by means of factor analysis. Rushinek and Rushinek regard factor analysis as a process of identifying variables which are highly correlated and somewhat redundant and suggesting new independent variables or factors to replace the original ones[Rushinek and Rushinek, 1983]. Table 32 shows the partial correlation matrix and terminal factors which result from the SAS Factor Analysis. Notice the high measures of correlation and that the six variables are reduced to two factors. The results indicate that the information content of the six variables may possibly be provided by two

Table 30. Regrouping College Education, Major and Conversion Categories.

| New Education Types | Frequency | Percent | Previous Categories |
|---------------------------|-----------|---------|----------------------|
| None | 141 | 43.4 | None |
| Some College | 93 | 28.6 | Some College |
| | | | Associate's |
| | | | Post Associate's |
| Graduate | 91 | 28.0 | Bachelor's and |
| | | | Higher |
| | 325 | | |
| New Major Types | | | |
| Computer Science | 49 | 15.1 | Computer Science |
| DP-MIS/Math/Science | 49 | 15.1 | Business DP/MIS |
| | | | Phy/Bio Sciences |
| | | | Math/Engineering |
| Other | 52 | 16.0 | Social Science |
| | | | Management and Other |
| None | 175 | 53.8 | None |
| | 325 | | |
| New Conversion Exp. Types | | | |
| No Experience | 204 | 62.8 | No Experience |
| | | | System/Equip. Know. |
| Some Experience | 55 | 16.9 | Other Exp-Some/Lit. |
| Greater Experience | 18 | 5.5 | Other Exp-Consid. |
| | | | COBOL 68 to 74 Exp |
| Early Phase IV Experience | 48 | 14.8 | Early Phase IV Exp |
| | 325 | | |

Table 31. Summary of Chi-Square Tests.

Null Hypothesis(Ho) for test: Variables are Independent.
Alpha Level Used was 0.10.

| Variables Tested | Results of Test | Prob Type I Error | Association |
|------------------|-------------------|-------------------|-------------|
| MAJOR & DEGREE | Reject Ho | 0.0001 | 0.668 |
| MAJOR & PTYPE | Can Not Reject Ho | 0.5072 | 0.158 |
| DEGREE & PTYPE | Can Not Reject Ho | 0.3860 | 0.138 |
| DEGREE & CONEXP | Can Not Reject Ho | 0.1473 | 0.169 |
| MAJOR & CONEXP | Can Not Reject Ho | 0.3603 | 0.172 |
| CONEXP & PTYPE | Reject Ho* | 0.0001 | 0.327 |
| MAJOR & FTNG6 | Reject Ho* | 0.0314 | 0.163 |
| DEGREE & FTNG4 | Reject Ho | 0.0480 | 0.135 |
| DEGREE & FTNG1 | Reject Ho | 0.0536 | 0.133 |
| PTYPE & FTNG5 | Reject Ho | 0.0023 | 0.207 |
| PTYPE & FTNG1 | Reject Ho | 0.0262 | 0.166 |
| CONEXP & FTNG5 | Reject Ho | 0.0519 | 0.152 |
| CONEXP & FTNG2 | Reject Ho | 0.0208 | 0.171 |
| CONEXP & FTNG1 | Reject Ho | 0.0325 | 0.162 |

All paired tests not shown resulted in No Rejection.

Most Training(FTNG vs FTNG) tests resulted in Rejection.

* Table is so sparse that Chi-Square test may not be valid.

Table 32. Partial Correlation Matrix and Factor Analysis.

| Correlation Matrix | | | | | |
|--------------------|---------|---------|---------|----------|---------|
| | TOTEXP | PGMEXP | C68EXP | C74EXP | JCLEXP |
| TOTEXP | 1.00000 | 0.76981 | 0.69660 | -0.00316 | 0.21817 |
| PGMEXP | | 1.00000 | 0.88327 | 0.09837 | 0.24922 |
| C68EXP | | | 1.00000 | -0.06478 | 0.22330 |
| C74EXP | | | | 1.00000 | 0.20803 |
| JCLEXP | | | | | 1.00000 |

| Factor Analysis Pattern | | |
|-------------------------|---------|----------|
| | FACTOR1 | FACTOR2 |
| TOTEXP | 0.87365 | -0.11573 |
| PGMEXP | 0.95173 | -0.03793 |
| C68EXP | 0.91842 | -0.17792 |
| C74EXP | 0.06721 | 0.85014 |
| JCLEXP | 0.38792 | 0.62766 |

variables or factors. The first factor primarily contains the variables of programmer experience (PGMEXP), total years of experience in the computer field (TOTEXP) and years of COBOL-68 experience (C68EXP). The second factor is mostly a composite of years of COBOL-74 experience (C74EXP) and years of JCL experience (JCLEXP).

Finally, Table 33 presents a summary of the programmers by experience levels. Also shown in the table are averages for all the years of experience in each area.

Program Information Data Analysis

When a Phase IV organization completes the conversion and testing of a program, a corresponding program information form is submitted to

Table 33. Phase IV Programmers Experience Summary.

| Experience Level of PGMEXP | Frequency | Percent |
|--|-----------|---------|
| Trainee (1 or less years exp.) | 93 | 28.6 |
| Intermediate (2 to 3 years exp.) | 78 | 24.0 |
| Experienced (4 to 6 years exp.) | 54 | 16.6 |
| Senior (7 or more years exp.) | 100 | 30.8 |
| Average Years of Experience In All Areas | | |
| Total Field Experience(TOTEXP): | 9.68 | |
| Programming Experience(PGMEXP): | 5.93 | |
| COBOL 68 Experience(C68EXP): | 4.19 | |
| COBOL 74 Experience(C74EXP): | 0.44 | |
| JCL Experience(JCLEXP): | 1.59 | |

the AFASPO. One of the centers, the Air Force Data Systems Design Center (AFDSDC), has been authorized to report their conversion progress by using an alternate method. The AFDSDC typically reports every two weeks with general progress information and periodically provides a man-hour report by system, not by program, without detailing the programmers involved. These system level man-hour counts were used during the cost estimation analysis presented in Chapter 6.

The data collection for the programmer productivity analysis was stopped when a sample of 130 programs was accumulated. A presentation format similar to that for programmer resume records is used to provide the essential facts.

The first basic summary is that of the number of programs by type and number of programmers assigned. Table 34 shows that the sample includes only COBOL programs and only two programs had online requirements.

Table 34. Phase IV Programs By Type and Number of Programmers.

| Number of Programmers | Number of Programs | Type | |
|-------------------------------------|-----------------------|-------|--------|
| | | Batch | Online |
| 1 | 51 | 49 | 2 |
| 2 | 24 | 24 | |
| 3 | 22 | 22 | |
| 4 | 20 | 20 | |
| 5 | 7 | 7 | |
| 6 | 3 | 3 | |
| 7 | 3 | 3 | |
| | 130 | 128 | 2 |
| *All Programs are COBOL-68 Programs | | | |

In order to have a supplementary difficulty measure, a count of the number of difficulty categories checked was calculated. Table 35 presents the difficulty profile of the programs. The first part of the table summarizes the counts of each of the difficulty categories as individually checked by a programmer. The second part shows the number of programs for each total count of difficulties checked per program.

Table 36 summarizes the average number of hours spent on each of the six conversion activities stipulated on the program information form activity matrix. The table also shows the average percentages of effort expended for each activity. It was discovered that most of

Table 35. Program Difficulty Counts & Totals.

| Difficulty | Count | Difficulty | Count |
|--------------------|--------------------|-------------|-------|
| SORT | 80 | SWITCHES | 55 |
| ZIP | 50 | COMP DATA | 49 |
| COPY | 26 | INTERROGATE | 21 |
| RANDOM I/O | 24 | REEL #'S | 20 |
| KALL | 13 | | |
| Total Difficulty | | | |
| Checks Per Program | Number of Programs | Percent | |
| 0 | 20 | 15.4 | |
| 1 | 37 | 28.5 | |
| 2 | 15 | 11.5 | |
| 3 | 16 | 12.3 | |
| 4 | 13 | 10.0 | |
| 5 | 8 | 6.1 | |
| 6 | 17 | 13.1 | |
| 7 | 3 | 2.3 | |
| 8 | 1 | 0.8 | |

Table 36. Conversion Activities: Times & Percentages.

| Activity | Avg Time | Percent |
|-------------------------|----------|---------|
| Documentation | 2.93 | 4.6 |
| Data File Transfer | 3.76 | 5.8 |
| ADS Translation | 14.85 | 23.1 |
| Create Control Language | 5.33 | 8.3 |
| Test/Debug | 28.41 | 44.2 |
| Miscellaneous | 9.06 | 14.1 |
| | 64.34 | |

the data reported contained no count of man-hours for the first two activities or categories: documentation and data file transfer. The reason for this was that these two activities were typically handled

at a system level and the accumulated man-hours were either not reported at all or recorded on the activity matrix of one program in the system. The impact of this reporting discrepancy was significant only in that the total effort hours for productivity analysis were derived from summing only four of the six categories of the activity matrix. This increased the overall uniformity of the data for the study. A collection of productivity and other measures which shows the overall condition and content of the information file is provided in Table 37.

Table 37. Productivity and Other Summary Measures.

| Average Measures | Mean | Min | Max |
|--|------|------|-------|
| Lines of Code Per Hour(LOCPEHR): | 44.4 | 3.53 | 396.8 |
| Hours Per Hundred Lines of Code(HRPERHLO): | 6.0 | 0.25 | 28.3 |
| Total Hours Per Program(excluding documentation and data file transfer): | 57.7 | 2.30 | 368.0 |
| Starting Lines of Code(STLOC): | 1175 | 121 | 4650 |
| Finishing Lines of Code(FILOLOC): | 1260 | 149 | 4849 |

To allow another view of the data, a file subset was created which contained those programs assigned to only one programmer. Using this file subset, a SAS generated cubic graph of the relationship between lines of code per hour (LOCPEHR) and the programmer's knowledge of the program (KCA) was produced and is shown in Figure 5. Notice that productivity decreases as the programmer's knowledge of the program increases. This counter-intuitive behavior of the Phase IV data supports a statement by Oliver that programmers

converting their own programs may not resist the temptation to "improve" the programs they are converting[Oliver, 1978]. No conclusions may be drawn about other conversion environments and this phenomenon may change as the Phase IV conversion effort continues. This manifestation is discussed further in chapter 5.

The variable LOCPERHR was similarly plotted against the count of difficulty(SUMDIF) using SAS and the results are shown in Figure 6. As expected, productivity generally decreases with an increase in the difficulty of the program. The initial dip or downward slope at SUMDIF=0 indicates that there are some complexity items not included on the Phase IV program information form for the programmers to check or some programmers failed to properly identify the difficulty categories for their programs. Discounting the initial dip of the curve, the decreasing productivity as the the program difficulty increases is likely to be true of all conversion efforts.

SAS

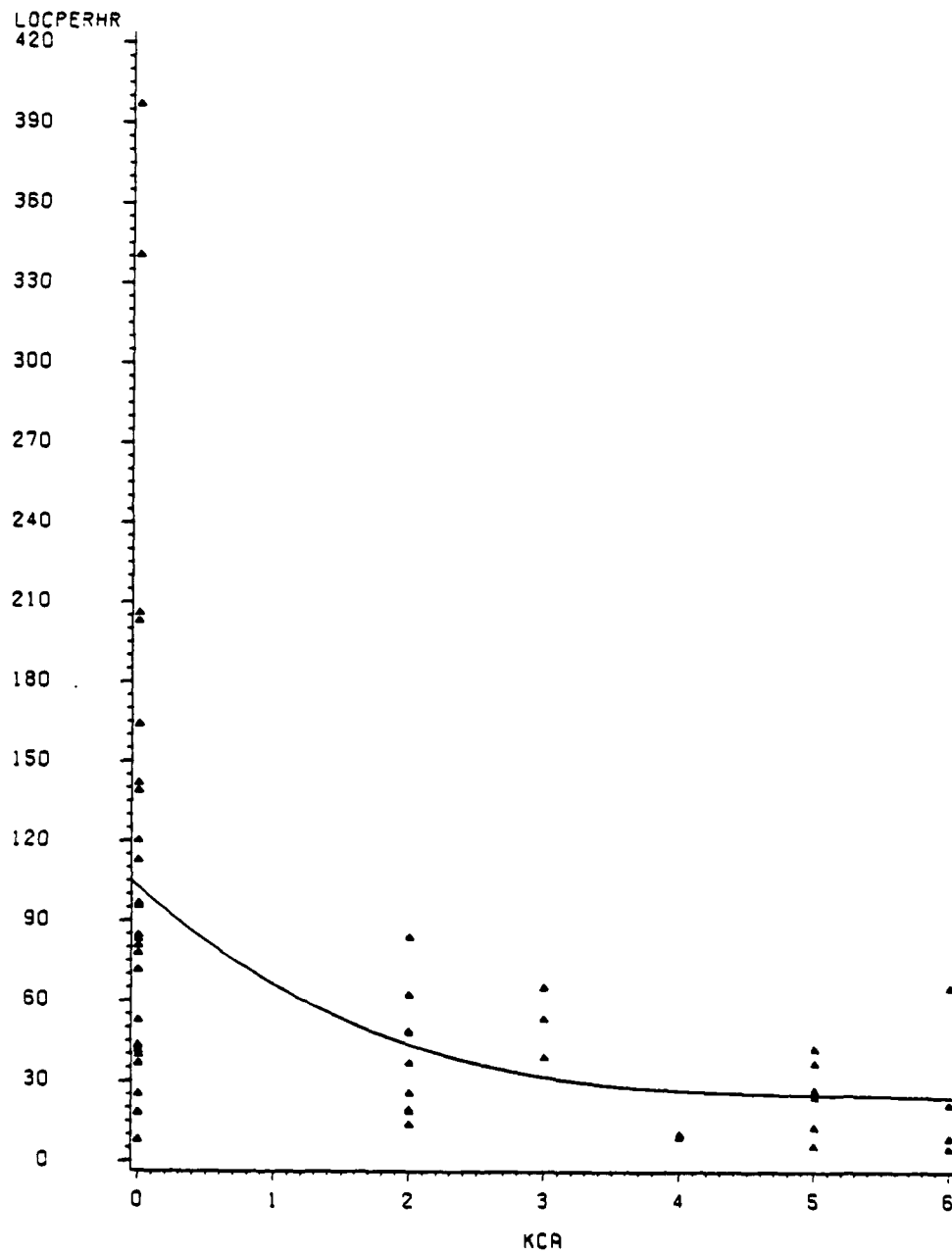


Figure 5. LOC PER HR versus Knowledge(KCA) of programmer for single programmer type programs.

SAS

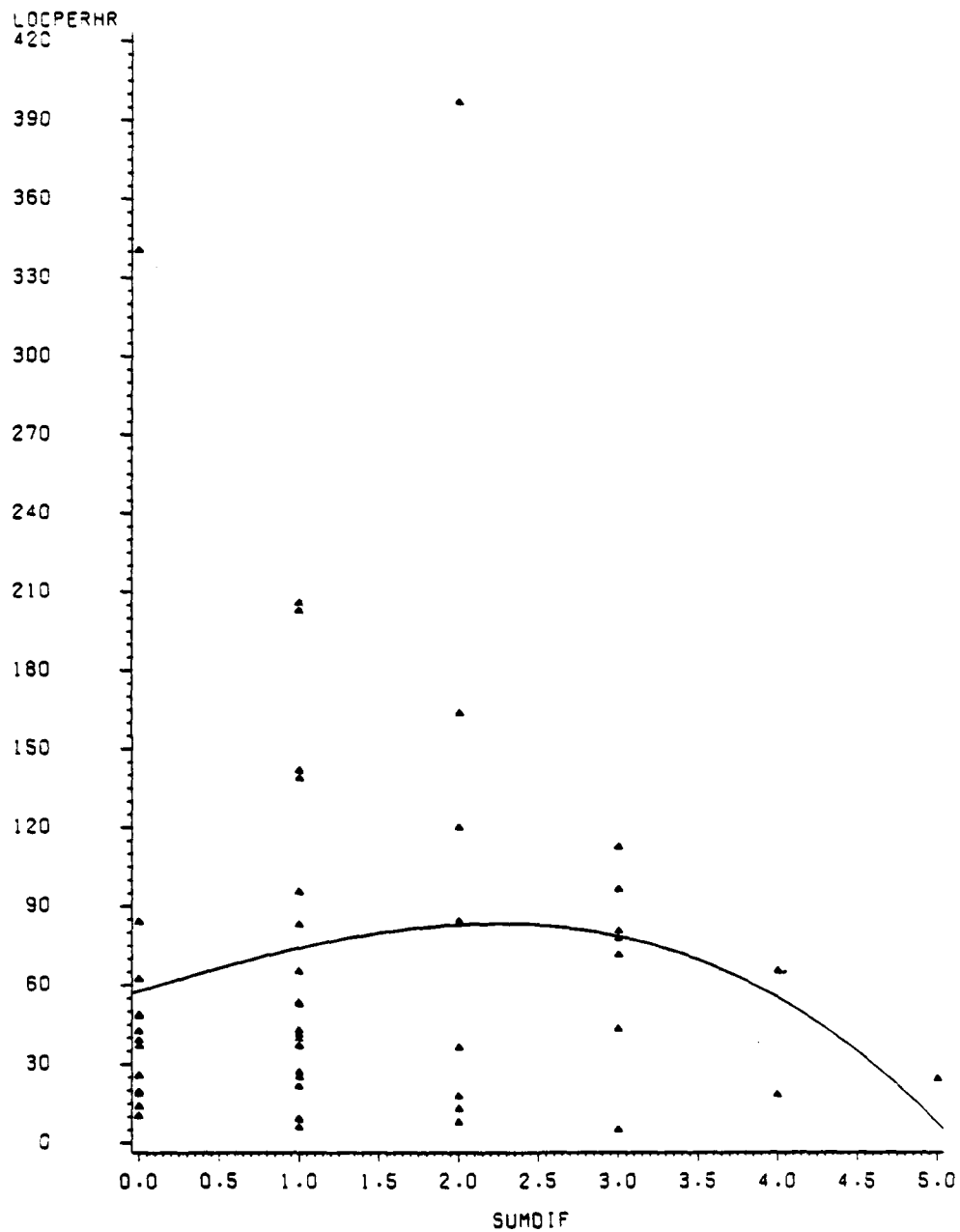


Figure 6. LOC PER HR versus Program Difficulty(SUMDIF) for single programmer type programs.

APPENDIX C

PROGRAMMER RESUME FILE

This appendix contains the programmer resume file in its entirety. For a description of the variables refer to Appendix B. The SAS listing of the programmer resume records follows.

PROGRAMMER RESUME FILE

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|
| U | C | L | E | X | P | 0 | 2 | 4 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 5 | 3 | 2 | 0 | 3 | 5 | 3 | 1 | | | | | |
| C | D | N | E | X | P | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 3 | 0 | 0 | 3 | 2 | 0 | 0 | 2 | 2 | 4 | 0 | 0 | 2 | 4 | 0 | 0 | | | |
| S | Y | S | E | X | P | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | |
| C | D | N | L | A | N | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | | | |
| P | T | Y | P | E | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 3 | 0 | 3 | 2 | 2 | 2 | 2 | 2 | 0 | 2 | 2 | 0 | 3 | 2 | 2 | 0 | 0 | 2 | 2 | 1 | 0 | 2 | 2 | 3 | 2 | | | |
| C | 7 | 4 | E | X | P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| C | 6 | 8 | E | X | P | 16 | 7 | 2 | 8 | 2 | 14 | 14 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 3 | 1 | 13 | 13 | 2 | 2 | 0 | 1 | 14 | 0 | 5 | 14 | 13 | 0 | 0 | 0 | 9 | 4 | 1 | 0 | 15 | 9 | 0 | 0 | | |
| P | G | M | E | X | P | 16 | 14 | 2 | 8 | 2 | 18 | 16 | 1 | 2 | 2 | 1 | 3 | 1 | 0 | 3 | 2 | 14 | 15 | 2 | 2 | 0 | 2 | 24 | 0 | 5 | 20 | 23 | 1 | 0 | 0 | 0 | 18 | 4 | 1 | 0 | 21 | 13 | 6 | 3 | | |
| T | O | T | E | X | P | 19 | 19 | 4 | 18 | 17 | 18 | 19 | 3 | 2 | 17 | 17 | 15 | 20 | 22 | 6 | 6 | 20 | 19 | 13 | 5 | 3 | 24 | 1 | 19 | 20 | 23 | 3 | 15 | 25 | 18 | 4 | 1 | 2 | 0 | 25 | 16 | 6 | 3 | | | |
| F | T | N | G | 1 | | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | | |
| F | T | N | G | 2 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | |
| F | T | N | G | 3 | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | |
| F | T | N | G | 4 | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | |
| F | T | N | G | 5 | | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | | |
| F | T | N | G | 6 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| M | I | N | O | R | | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 0 | 1 | 0 | 2 | 2 | 2 | 2 | 2 | 0 | 5 | | | | |
| M | A | J | J | O | R | 1 | 0 | 0 | 0 | 0 | 0 | 6 | 5 | 0 | 0 | 0 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 4 | 1 | 0 | 2 | 0 | 2 | 2 | 5 | 0 | 5 | 4 | 2 |
| D | E | G | R | E | E | 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 | 4 | 0 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | | | |
| P | G | M | I | D | | E | F | H | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | X | Y | Z | A | B | C | D | E | F | G | H | I | | | | |
| M | A | J | C | O | M | F | F | F | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | G | H | H | H | H | H | H | H | H | H | H | | | | | |
| O | B | S | | | | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | | | | |

PROGRAMMER RESUME FILE

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|
| J | C | L | E | X | P | 4 | 0 | 2 | 5 | 1 | 1 | 5 | 5 | 5 | 5 | 0 | 2 | 5 | 4 | 10 | 9 | 1 | 1 | 1 | 1 | 1 | 5 | 0 | 0 | 0 | 1 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 5 | 1 | 0 | 1 | 10 | |
| C | O | N | E | X | P | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 3 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 4 | 4 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | |
| S | Y | S | E | X | P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| C | O | N | L | A | N | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | | | |
| P | T | I | P | E | 3 | 1 | 1 | 1 | 2 | 2 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 2 | 3 | 3 | 3 | 4 | 0 | 0 | 1 | 1 | 3 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 3 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 3 | | |
| C | 7 | 4 | E | X | P | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| C | 6 | 8 | E | X | P | 3 | 1 | 1 | 16 | 16 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 5 | 2 | 10 | 15 | 0 | 0 | 3 | 3 | 2 | 11 | 3 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 3 | 1 | 2 | 1 | 1 | 2 | 16 |
| P | G | M | E | X | P | 4 | 1 | 1 | 16 | 19 | 3 | 0 | 0 | 11 | 2 | 0 | 0 | 7 | 3 | 10 | 20 | 21 | 2 | 2 | 3 | 2 | 12 | 3 | 1 | 1 | 3 | 1 | 1 | 5 | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 3 | 3 | 1 | 2 | 2 | 20 | |
| T | O | T | E | X | P | 4 | 12 | 11 | 11 | 16 | 19 | 4 | 0 | 26 | 2 | 0 | 0 | 20 | 4 | 13 | 29 | 22 | 4 | 6 | 9 | 14 | 10 | 5 | 1 | 1 | 3 | 1 | 1 | 5 | 1 | 1 | 0 | 0 | 0 | 0 | 4 | 3 | 17 | 6 | 4 | 2 | 26 | |
| F | T | N | G | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | |
| F | T | N | G | 2 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | |
| F | T | N | G | 3 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | |
| F | T | N | G | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| F | T | N | G | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | |
| F | T | N | G | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| M | I | N | Q | R | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| M | A | J | Q | R | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| D | E | G | R | E | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| P | G | M | I | D | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | AA | AB | AC | AD | AE | AF | AG | AH | AI | AJ | AK | AL | AM | AN | AO | AP | AQ | AR | AS | AT | AU | AV | AW | AX | AY | AZ | | |
| M | A | J | C | M | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | H | K | | |
| O | B | S | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | | | | | | | | | |

PROGRAMMER RESUME FILE

| | |
|---------|---|
| COLEXP | 010000000310100005013100010101101110003030 |
| CONEXP | 0400000040000004044020000000044233040000044 |
| SYSEXPR | 000 |
| CONLAN | 44 |
| PTYPE | 01012211232000222332330333022123312333122 |
| G74EXP | 0100000100000000100000000000000000000000000 |
| G6BEXP | 01013322311000101233314100003315441101201023 |
| PGMEXP | 01013322411000101613331600011033154411171222101043 |
| TDTEXP | 45111332222112202217832000021025131513172022181810203 |
| FTNG1 | 111000000000000000000000000000000000000000 |
| FTNG2 | 11 |
| FTNG3 | 11 |
| FTNG4 | 00 |
| FTNG5 | 00 |
| FTNG6 | 00 |
| MINOR | 040000002000000000000000000000000000000000 |
| MAJOR | 060000506000002000000000000000000000000000 |
| DEGREE | 020000201000060000000000000000000000000000 |
| PGMID | F G H I J K L M N O P Q R S T U V W X Y Z 1 2 A B C D E F |
| MAJCOM | T |
| OB S | 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 |

PROGRAMMER RESUME FILE

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------------|---|---|----|---|---|---|----|----|---|---|---|---|---|---|----|----|----|----|----|---|----|----|----|----|---|----|----|----|----|---|----|----|
| U C L E X P | 0 | 4 | 0 | 1 | 2 | 4 | 4 | 4 | 3 | 0 | 2 | 0 | 0 | 1 | 0 | 9 | 11 | 12 | 1 | 1 | 10 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 1 | 1 | 0 | 5 |
| C O M E X P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 1 | 2 | 2 | 0 | 0 | 0 | 4 | 4 | 2 | 0 | 0 | 5 | 0 | 0 |
| S Y S E X P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | |
| C O N L A N | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | |
| P T V P E | 3 | 1 | 1 | 2 | 1 | 2 | 3 | 4 | 0 | 2 | 3 | 2 | 3 | 0 | 2 | 2 | 2 | 3 | 3 | 0 | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 3 | 2 |
| C 7 4 E X P | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 6 | 3 | 0 | 1 | 0 | 0 | 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| C 6 8 E X P | 1 | 8 | 4 | 2 | 0 | 1 | 0 | 8 | 0 | 2 | 2 | 1 | 1 | 0 | 0 | 4 | 12 | 3 | 3 | 0 | 0 | 14 | 0 | 7 | 3 | 2 | 2 | 10 | 14 | 4 | 1 | 10 |
| P G M E X P | 1 | 6 | 8 | 3 | 1 | 1 | 2 | 10 | 3 | 3 | 2 | 1 | 1 | 0 | 0 | 18 | 25 | 9 | 5 | 3 | 1 | 8 | 26 | 8 | 7 | 3 | 2 | 10 | 14 | 4 | 1 | 15 |
| T D T E X P | 1 | 8 | 18 | 4 | 1 | 1 | 24 | 3 | 3 | 4 | 1 | 1 | 0 | 0 | 23 | 25 | 15 | 13 | 15 | 2 | 12 | 36 | 9 | 13 | 3 | 15 | 10 | 14 | 4 | 2 | 16 | |
| F T T N G 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | |
| F T T N G 2 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | |
| F T T N G 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| F T T N G 4 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F T T N G 5 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| F T T N G 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| M I N Q R | 0 | 1 | 0 | 0 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 4 | 4 | 0 | 4 | 4 | 3 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 2 |
| M A J Q R | 0 | 5 | 0 | 0 | 0 | 1 | 6 | 0 | 3 | 4 | 6 | 4 | 4 | 0 | 0 | 0 | 2 | 6 | 6 | 6 | 1 | 1 | 2 | 2 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 6 |
| D E G R E E | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 4 | 1 | 1 | 4 | 4 | 0 | 0 | 4 | 4 | 3 | 4 | 6 | 4 | 4 | 4 | 4 | 4 | 1 | 2 | 4 | 0 | 0 | 0 | 4 |
| P G M I D | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| M A J C O M | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Z | Z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |

O B S

297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325

APPENDIX D

PROGRAM INFORMATION FILE

This appendix contains the program information file in its entirety. Appendix B contains a description of the variables. The SAS listing of the program information records follows.

PROGRAM INFORMATION FILE

[illegible][illegible][illegible]

PROGRAM INFORMATION FILE

[illegible][illegible][illegible]

VITA

Major John Diego Fernandez, U.S. Air Force, was born on 13 August 1947, in Corpus Christi, Texas, the son of Mr. and Mrs. Fidel F. Fernandez. He attended Texas A&I University and received a B.A. degree in Mathematics in 1968. Prior to entering the Air Force, he was employed as a mathematician-statistician at Kelly AFB, San Antonio, Texas. Major Fernandez received his military commission in May 1969 and has served in various capacities since then. He began his career as a Communications-Electronics Maintenance Officer and after receiving his M.S.E. degree in Industrial Engineering from West Virginia University he commenced working in the computer field. While assigned to the Pentagon, Major Fernandez served as programmer and systems analyst in support of the Joint Chiefs of Staff. He was subsequently assigned overseas as an Exchange Officer with the Venezuelan Air Force where he served in the capacity of Chief, Systems Analysis Division. His last assignment was with the San Antonio Data Services Center where he worked as a Telecommunications Specialist and Chief, Customer Assistance Division. His decorations include the Meritorious Service Medal with two Oak Leaf Clusters, the Joint Service Commendation Medal and the Venezuelan Air Force Commendation Medal. He is a member of Alpha Pi Mu, Upsilon Pi Epsilon, the Computer Society of IEEE, and ACM. His permanent address is: 906 Nineteenth Street, Corpus Christi, Texas, 77840.

DATE
ILME